Optimisation de réseaux de capteurs sans fil pour le suivi de cibles mobiles

Charly Lersteau

UNIVERSITE
BRETAGNE
LOIRE

THÈSE / UNIVERSITÉ BRETAGNE-SUD
sous le sceau de l'Université Bretagne-Loire

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITÉ BRETAGNE-SUD

Mention : STIC
École doctorale : SICMA

Présentée par
Charly Lersteau

Préparée à l'unité mixte de recherche
(n° 6285)
Laboratoire des Sciences et Techniques de l'Information,
de la Communication et de la Connaissance

Optimisation de réseaux de capteurs sans fil pour le suivi de cibles mobiles

Thèse soutenue le 20 septembre 2016
devant le jury composé de :

M. François Clautiaux
Professeur des universités, Université de Bordeaux / Rapporteur
M. Patrick Jaillet
Professeur des universités, MIT, États-Unis / Rapporteur
M. Abdellatif Chafik
Professeur des universités, Institut supérieur du Génie Appliqué, Casablanca / Examinateur
M. Xavier Delorme
Maître-assistant, École des Mines de Saint-Étienne / Examinateur
M. Giovanni Felici
Professeur des universités, Consiglio Nazionale delle Ricerche, Rome / Examinateur
M. Daniel Vanderpooten
Professeur des universités, Université Paris-Dauphine / Examinateur
M. Marc Sevaux
Professeur des universités, Université Bretagne-Sud / Directeur de thèse
M. André Rossi
Professeur des universités, Université d'Angers / Co-directeur de thèse

Résumé

Les réseaux de capteurs sans fil suscitent une attention croissante depuis quelques années, tant les applications sont nombreuses, incluant notamment le suivi de véhicules ou la surveillance de champs de bataille. Un ensemble de capteurs disséminé aléatoirement a pour but de surveiller des cibles se déplaçant dans une région donnée. Chaque capteur a une durée de vie limitée et deux états : actif ou inactif. Un capteur actif peut surveiller des cibles dans son rayon de portée, au prix d'une consommation d'énergie. Dans cette thèse, les problèmes étudiés consistent à déterminer un ordonnancement optimal d'activités de surveillance, afin de couvrir toutes les cibles à tout instant de la mission.

Nous abordons en premier lieu un problème d'ordonnancement robuste. Une cible dont on connaît la trajectoire spatiale avec précision est sujette à incertitude temporelle. Cette situation est rencontrée lorsqu'un avion vole dans un couloir aérien, qu'un train circule sur une voie ferrée, ou que de tout autre véhicule suit un itinéraire pré-déterminé. L'objectif est de calculer un ordonnancement d'activités capable de résister au plus grand écart par rapport aux dates prévisionnelles de passage de la cible. Ce premier problème est résolu à l'aide d'un algorithme exact pseudo-polynomial, reposant sur une dichotomie.

En second lieu, nous étudions le problème visant à préserver la capacité de surveillance du réseau de capteurs dans un contexte multi-missions. Les cibles sont maintenant sujettes à une incertitude spatiale, c'est-à-dire susceptibles de se trouver à une distance inférieure à un seuil δ de leur position prévisionnelle. Ce second problème est résolu à l'aide d'un algorithme exact basé sur la génération de colonnes, et accéléré par une métaheuristique.

Les méthodes de résolution proposées ont en commun une étape préliminaire, appelée discrétisation, qui conduit à reformuler les problèmes originaux comme des problèmes d'ordonnancement d'activités de surveillance. L'espace de surveillance est découpé en faces, ensembles de points couverts par un même sous-ensemble de capteurs. Le calcul des durées de séjour des cibles dans chaque face permet de découper la durée de la mission en fenêtres de temps, et d'envisager le problème de couverture de cibles mobiles comme une séquence de problèmes de couverture de cibles immobiles.

Les algorithmes proposés pour aborder ces problèmes sont testés sur de nombreuses instances, et leurs résultats sont analysés. De nombreuses perspectives ouvertes par ces travaux sont également présentées.

Abstract

Wireless sensor networks have received a particular attention during the last years, involving many applications, such as vehicle tracking or battlefield monitoring. A set of sensors is randomly dispatched in a region in order to monitor moving targets. Each sensor has a limited battery lifetime and two states: active or inactive. An active sensor is able to monitor targets inside its sensing radius, which consumes energy. In this thesis, the studied problems consist in deciding an optimal schedule of sensing activities, in order to cover all the targets at any instant of the mission.

First, we study a robust scheduling problem. A target such that the spatial trajectory is exactly known is subject to temporal uncertainties. This context is met for a plane flying in an airline route, a train running on a railway, or any vehicle following a predetermined path. The objective is to compute a schedule of activities able to resist to the largest uncertainties. This first problem is solved using an exact pseudo-polynomial algorithm, relying on a dichotomy.

Second, we study a problem aiming at preserving enough sensor network capacity in order to perform further missions. For this problem, the targets are subject to spatial uncertainties, i.e. their actual position may be at a distance δ of their expected position. This second problem is solved using an exact algorithm based on column generation, accelerated by a metaheuristic.

All the proposed methods have a common phase, called discretization, that leads to reformulate the original problems as activity scheduling problems. The monitored area is split into faces, that are defined as sets of points covered by the same set of sensors. Computing the stay duration of targets inside each face leads to split the mission duration into time windows, so the moving target tracking problem can be seen as a sequence of static target tracking problems.

The proposed algorithms are tested on many instances, and the analysis of the results is provided. Numerous open perspectives of this work are also given.

Remerciements

Le travail réalisé dans cette thèse de doctorat n'aurait probablement pas été possible sans le soutien d'un certain nombre de personnes que je tiens à remercier, pour leur bienveillance, leur intérêt, leur bonne humeur et leur générosité.

J'aimerais en premier lieu remercier chaleureusement mes directeurs de thèse, Marc Sevaux et André Rossi, pour la confiance qu'ils m'ont accordée en me permettant de réaliser cette thèse, et pour m'avoir brillamment guidé et suivi avec grande attention pendant ces trois années.

Je tiens également à remercier mes plus chers collègues. Fabian, qui a toujours amené la bonne humeur au laboratoire. Hugues, qui m'a soutenu avec grande attention depuis le début. Erwan, qui apporte toujours sa touche d'humour et dérision pour nous faire sourire. Maria, une amie très agréable avec qui j'aime discuter. Asma, très brillante et d'une gentillesse sans pareille. Mourad, très amical et très sympathique. Claudia, une amie extrêmement chaleureuse. Je remercie également Audrey et Alban, Johann, Robin, Vincent, Alexandre, Valère, Alexis, Antoine, Mikael, Pierre, Ahmed, Clément, Clara, Borja, José de Jesus, Emmanuel, Trung, Raje, Abhishek, Maxime, Maria, Gilberto, Delphine, Vianney, Mathilde, Ghizlane, Cédric, Kévin, Guy, Jean-Philippe, Dominique... Les secrétaires Florence et Virginie qui, avec un humour sans précédent, répandent la bonne humeur et illuminent nos journées au bureau. Des remerciements pour tout le reste des collègues du centre de recherche, de l'université, dont je n'ai pas cité les noms, qui ont été amicaux et coopératifs lorsque j'en avais besoin.

Mon parcours universitaire n'aurait sans doute pas abouti sans les professeurs du master Optimisation en Recherche Opérationnelle de l'Université de Nantes, que je remercie sincèrement, ainsi que mes très chers camarades, avec qui j'ai d'excellents souvenirs.

Enfin, je suis particulièrement redevable à mes parents, à ma famille, pour leur immense soutien moral et leur confiance indéfectible en mes projets.

Table des matières

Chapitre 1

Introduction

Contenu

1.1 Motivations

Les avancées technologiques en matière de production de circuits électroniques, et dans une moindre mesure, les progrès dans le domaine des batteries pour le stockage de l'énergie électrique, ont permis le développement de capteurs autonomes, de petite taille et dont le coût de fabrication est de plus en plus faible. Les capteurs sont des appareils capables de récolter différentes sortes d'informations dans leur rayon d'action et de communiquer avec d'autres capteurs. Les réseaux de capteurs sans fil sont une technologie très prometteuse et déjà convoitée. Et pour cause, ils ont été cités dans la revue *Technology Review* ("10 Emerging Technologies That Will Change the World" 2003), éditée par le MIT, dans la liste des innovations qui "changeront le monde". Selon la société d'analyses *Frost & Sullivan*, le marché des réseaux de capteurs sans fil estimé à 1,2 milliard de dollars en 2014, va

progresser en moyenne de 18,1% par an pour atteindre la valeur de 3,26 milliards de dollars en 2020 ("Wireless Sensor Networks See Uptake in Global Industries Due to Technological Breakthroughs" 2015).

De nouvelles applications sont ainsi rendues possibles à moindre coût, en particulier la surveillance des cibles. L'utilisation de réseaux de capteurs sans fil comporte de nombreux avantages. Ils sont *ad hoc*, c'est-à-dire qu'ils ne nécessitent pas d'infrastructure préalable et peuvent être largués dans l'urgence depuis un avion ou un hélicoptère, par exemple dans des zones ravagées par les catastrophes naturelles. Dans les applications militaires, le contexte est typiquement celui de la projection de forces armées en territoire hostile ou inconnu, où les infrastructures sont inadaptées, endommagées, voire inexistantes. Les cibles à suivre peuvent être des soldats, des véhicules terrestres et/ou des aéronefs, amis ou ennemis.

Un capteur est typiquement équipé d'un ou plusieurs dispositifs de mesure ou d'acquisition de données (mouvements, température, humidité, lumière...), d'un dispositif de traitement numérique de ces données (processeur), d'une unité de communication (qui permet aux capteurs de transmettre des données aux capteurs voisins ou une station de base dans un rayon de portée donné), et d'une unité de stockage d'énergie (batterie) (Yick *et al.* 2008). Les capteurs sont par conséquent des appareils peu complexes et faciles à configurer. Leur faible coût permet ainsi de déployer un grand nombre de capteurs afin de surveiller de vastes zones accidentées ou inaccessibles. Grâce au grand nombre de capteurs qui le constitue, un réseau de capteurs sans fil résiste plus facilement aux attaques d'un ennemi ou aux dysfonctionnements spontanés, ce qui le rend très approprié aux applications militaires.

Les multiples applications possibles ont engendré de nouvelles problématiques à résoudre. En effet, les capteurs étant autonomes et alimentés par une batterie, la consommation énergétique est considérée comme l'un des aspects les plus importants des réseaux de capteurs sans fil. De plus, le prélèvement d'énergie de l'environnement (typiquement par le biais de cellules photovoltaïques) n'est pas toujours possible pour des raisons d'encombrement, de discrétion et de coût, et les progrès en matière de stockage énergétique, bien que significatifs, n'ont pas eu la même ampleur que ceux qui ont permis la miniaturisation et la baisse des coûts de production des circuits électroniques (Zu *et al.* 2011). Cette thèse considère l'optimisation de l'utilisation des réseaux de capteurs sans fil en contexte multi-missions, c'est-à-dire que l'on souhaite optimiser l'usage du réseau de capteurs pour remplir la mission courante, tout en considérant que le réseau devra remplir des missions de surveillance ultérieures (Jishy 2011). L'incertitude sur le comportement des cibles à surveiller conduit à considérer des problèmes d'optimisation robuste, dont les solutions

permettent de suivre les cibles malgré des aléas. Dans certaines applications, l'emplacement des capteurs peut être décidé. Le problème de placement de capteurs peut se réduire au problème des gardiens de musée (Art Gallery Problem) (Meguerdichian *et al.* 2003). Toutefois, les problèmes de placement de capteurs ne seront pas abordés dans cette thèse, où l'on considère que la position des capteurs est une donnée du problème.

1.2 État de l'art

1.2.1 Problématiques des réseaux de capteurs

Depuis quelques années, de nombreux protocoles de surveillance et de communication ont été conçus afin de répondre à diverses problématiques. Chaque protocole est dédié à l'optimisation d'un ou plusieurs objectifs. Cette thèse n'abordera pas les protocoles en détail. Le lecteur est dirigé vers plusieurs états de l'art qui donnent une vue d'ensemble des protocoles (Demigha *et al.* 2013 ; Naderan *et al.* 2009, 2012 ; Wang *et al.* 2006). Cette section présente une brève liste non-exhaustive des principaux critères utilisés pour comparer les réseaux de capteurs sans fil.

La *consommation énergétique* est l'un des critères les plus importants, puisque les capteurs sont généralement équipés d'une batterie non-rechargeable. Il est par ailleurs l'un des plus étudiés et fait l'objet d'une littérature très abondante. L'un des protocoles phares est DCTC (Zhang *et al.* 2004), qui calcule successivement des arbres de coût minimum en suivant les déplacements d'une cible. Beaucoup de protocoles répondant à ce critère reposent sur LEACH (Handy *et al.* 2002) ou HEED (Younis *et al.* 2004). Une meilleure *précision de suivi* peut être atteinte en activant davantage de capteurs ou en prédisant correctement le comportement des cibles. Une bonne technique de prédiction des cibles peut notamment aider à activer moins de capteurs et économiser les batteries. Yang *et al.* (2003) et Xu *et al.* (2004) proposent des protocoles basés sur la prédiction. Les réseaux de capteurs pouvant comporter un très grand nombre de nœuds, les protocoles doivent pouvoir s'adapter à toutes tailles de réseaux. La *scalabilité* signifie la capacité à s'adapter à toutes tailles de réseaux. Ce critère est généralement crucial dans la conception de protocoles, afin d'éviter la surcharge des réseaux. Par exemple, Kung *et al.* (2003) se sont particulièrement penchés sur cet aspect. Dans le cas où les capteurs subissent des dysfonctionnements ou les attaques d'un ennemi, il peut être nécessaire d'anticiper les aléas. La *robustesse* ou *tolérance aux erreurs* répond à cette problématique et fait l'objet d'un intérêt grandissant ces dernières années (Mahmood *et al.* 2015 ; Oracevic *et al.* 2014).

1.2.2 Surveillance de cibles fixes

Le problème relatif à la surveillance de cibles ponctuelles fixes revêt une importance particulière, car le problème de surveillance de surfaces s'y ramène (Berman *et al.* 2004 ; Deng *et al.* 2012 ; Slijepcevic *et al.* 2001). Pour ce faire, la région à surveiller est découpée en un ensemble de surfaces disjointes, et des cibles ponctuelles fictives sont introduites dans chacune de ces surfaces. Lorsque les zones de couverture des capteurs sont des disques, le nombre de cibles ponctuelles nécessaires ne dépasse pas $m(m-1)+2$ où m est le nombre de capteurs (Berman *et al.* 2004). Ces travaux ont inspiré la méthode de discrétisation présentée dans le chapitre 2.

Maximisation de la durée de vie du réseau

Dans le contexte de surveillance de cibles fixes, l'un des problèmes les plus étudiés est la maximisation de la durée de vie du réseau. Nous abrégerons ce problème par MNL, pour Maximum Network Lifetime. La durée de vie du réseau est définie par le temps durant lequel toutes les cibles sont couvertes. La formulation de ce problème repose généralement sur la notion de *couverture* (Slijepcevic *et al.* 2001), c'est-à-dire un sous-ensemble de capteurs capables de couvrir toutes les cibles. Ainsi, à chaque instant, il n'est pas nécessaire d'activer tous les capteurs, mais seulement une couverture. Quand un capteur n'est pas activé, il ne consomme pas d'énergie. Par conséquent, MNL peut être vu comme un problème d'ordonnancement de couvertures, où l'on doit déterminer le temps d'activation de chacune d'entre elles.

Dans sa version de base, MNL peut être formulé par le programme linéaire en variables continues MP^{MNL} (Berman *et al.* 2004). Soient m le nombre de capteurs et q le nombre de couvertures. a_{ic} est une constante binaire égale à 1 si le capteur $i \in \{1,\dots,m\}$ appartient à la couverture $c \in \{1,\dots,q\}$, 0 sinon. b_i est la capacité initiale du capteur $i \in \{1,\dots,m\}$. Cette capacité s'exprime par la durée pendant laquelle le capteur peut être utilisé. Les variables de décision sont définies par t_c, durée d'activation de la couverture $c \in \{1,\dots,q\}$. La durée de vie du réseau à maximiser est ainsi déterminée par la somme des t_c. La contrainte (1.2) assure que la durée d'activation de chaque capteur ne dépasse pas sa capacité. Les variables duales associées à cette contrainte sont notées π_i. Ce programme linéaire constitue la formulation de base d'un large éventail de variantes de MNL (Cardei *et al.* 2005a,b ; Castaño *et al.* 2013 ; Castaño *et al.* 2014, 2015 ; Raiconi *et al.* 2011 ; Rossi *et al.* 2011, 2012, 2013 ; Singh *et al.* 2013a,b). L'ordonnancement est construit en

Modèle $\mathrm{MP}^{\mathrm{MNL}}$

$$\max \sum_{c=1}^{q} t_c \tag{1.1}$$

$$\text{s.t.} \sum_{c=1}^{q} a_{ic} t_c \leq b_i \qquad \forall i \in \{1, \ldots, m\} \ [\pi_i] \tag{1.2}$$

$$t_c \geq 0 \qquad \forall c \in \{1, \ldots, q\} \tag{1.3}$$

activant les couvertures à tour de rôle, dans un ordre arbitraire, en respectant les durées données par les variables t_c.

Le nombre maximum de couvertures possibles croît exponentiellement en fonction du nombre de capteurs et le problème de décision associé à MNL est prouvé NP-complet (Cardei *et al.* 2005a). Pour ces raisons, il est généralement inenvisageable d'énumérer toutes les couvertures. Toutefois, il est généralement suffisant de construire un petit sous-ensemble de couvertures à l'aide d'heuristiques pour obtenir de bonnes solutions. $\mathrm{MP}^{\mathrm{MNL}}$ a la structure d'un problème de packing à variables fractionnaires (Berman *et al.* 2004). Ainsi, une résolution approchée à l'aide de l'algorithme de Garg-Könemann (Garg *et al.* 1998) est envisageable lorsque le nombre de couvertures est très élevé.

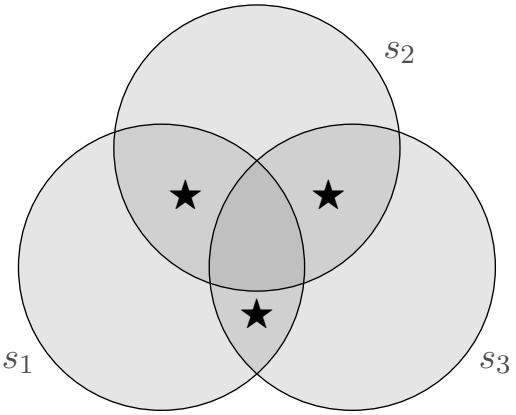


Figure 1.1 – Exemple avec 3 capteurs et 3 cibles (symbole ★)

Dans (Cardei *et al.* 2005c ; Slijepcevic *et al.* 2001), les couvertures sont disjointes, c'est-à-dire qu'un même capteur ne peut pas faire partie de plusieurs couvertures. Toutefois, en autorisant les couvertures non-disjointes, il s'ensuit que la durée de vie optimale du

réseau peut augmenter significativement (Berman *et al.* 2004 ; Cardei *et al.* 2005a,b). La figure 1.1 montre un exemple avec 3 capteurs, notés s_1, s_2, s_3, ayant une capacité de 2 unités de temps chacun, et 3 cibles. Dans le cas de couvertures disjointes, alors il n'est possible d'activer qu'une seule couverture durant 2 unités de temps, par exemple $(\{s_1, s_2\}, 2)$. Si on autorise les couvertures non-disjointes, l'ordonnancement $(\{s_1, s_2\}, 1)$, $(\{s_1, s_3\}, 1)$, $(\{s_2, s_3\}, 1)$ est réalisable et a une durée de 3 unités de temps.

Dans le cas particulier où la consommation énergétique des capteurs est proportionnelle au nombre de cibles surveillées, il existe un modèle linéaire en variables continues dont la taille est polynomiale en fonction du nombre de capteurs ou de cibles (Liu *et al.* 2011). Ce modèle prend en compte l'acheminement des données vers une station de base et les coûts de communication qui en découlent.

Méthodes heuristiques

De nombreuses méthodes heuristiques ont été proposées pour générer des couvertures. Slijepcevic *et al.* (2001) proposent un algorithme en $\mathcal{O}(m^2)$ qui construit des couvertures dont l'aire des surfaces les moins densément couvertes est minimisée. Cette heuristique génère au maximum k couvertures, où k est le nombre de capteurs couvrant la cible dite *critique*, c'est-à-dire la cible couverte par le moins de capteurs. Cardei *et al.* (2005c) génèrent les couvertures en exploitant la solution d'un problème de flot maximum, sur des graphes orientés bipartis, avec d'un côté les nœuds représentant les capteurs, et de l'autre, les cibles. Berman *et al.* (2004), après partitionnement de la zone de surveillance en surfaces disjointes, attribuent un coût à chaque surface, et un poids à chaque capteur, de telle sorte qu'un algorithme glouton crée des couvertures dont le ratio poids/coût est minimisé. Cardei *et al.* (2005a) calculent une contribution associée à chaque capteur en fonction de sa capacité et des cibles qu'il couvre, et sélectionnent en priorité les capteurs de meilleure contribution qui couvrent les cibles critiques. Delicato *et al.* (2006) formulent la génération de couvertures comme un problème de sac-à-dos, dont les profits sont calculés en fonction d'une utilité et de l'énergie résiduelle des capteurs, et dont la capacité représente une consommation d'énergie maximale autorisée. Wang *et al.* (2008) présentent une formulation de MNL sous la forme d'un programme linéaire en variables mixtes, et proposent deux algorithmes, l'un basé sur la relaxation linéaire et l'autre glouton. Dans (Cardei *et al.* 2005b), les capteurs peuvent ajuster leur rayon de portée pour modifier leur consommation énergétique. Parmi les heuristiques proposées, l'une se base sur la relaxation d'un programme linéaire en nombres entiers. Les autres sont deux versions d'un

algorithme glouton, l'une centralisée et l'autre distribuée, reposant sur la notion de contribution d'un couple (capteur, rayon de portée), qui sélectionnent les couples sur la base de cette contribution, tout en tenant compte des capacités restantes des capteurs. Dans (Yang *et al.* 2010), les capteurs sont directionnels, c'est-à-dire qu'ils couvrent un secteur angulaire plutôt qu'un disque. Les deux heuristiques présentées reposent sur l'idée d'attribuer des utilités aux capteurs et aux directions en fonction des distances avec les cibles.

Méthodes à base de génération de colonnes

Les algorithmes à base de génération de colonnes sont particulièrement appropriés pour résoudre MNL de manière exacte. En effet, MNL se prête bien à une décomposition de Dantzig-Wolfe (Dantzig *et al.* 1960) dans laquelle MP^{MNL} est la formulation d'un problème maître. Le problème de génération de couvertures est alors formulé comme un problème auxiliaire (Alfieri *et al.* 2007 ; Rossi *et al.* 2010) aussi appelé problème de pricing, ou encore sous-problème. Une formulation de base du problème auxiliaire, sous forme de programme linéaire en variables binaires noté PP^{MNL}, est proposée ci-dessous.

$$\text{PP}^{\text{MNL}} \begin{cases} \max\ 1 - \displaystyle\sum_{i=1}^{m} \pi_i a_{ic} & & (1.4) \\ \text{s.t.} \displaystyle\sum_{i \in S(j)} a_{ic} \geq 1 & \forall j \in \{1, \ldots, n\} & (1.5) \\ a_{ic} \in \{0, 1\} & \forall i \in \{1, \ldots, m\} & (1.6) \end{cases}$$

où m est le nombre de capteurs, n le nombre de cibles, a_{ic} une variable binaire égale à 1 si le capteur est sélectionné dans la couverture, 0 sinon, et $S(j)$ l'ensemble des capteurs capables de couvrir la cible j, que l'on désignera sous le terme de *capteurs candidats*. L'objectif, exprimé en fonction des variables duales de MP^{MNL}, consiste à maximiser le coût réduit de la nouvelle couverture. La contrainte (1.5) assure que toutes les cibles sont couvertes par au moins un capteur de la couverture. Notons que ce programme linéaire en variables binaires est similaire à la formulation d'un problème de couverture d'ensembles (Set Covering Problem).

L'idée sous-jacente des algorithmes de génération de colonnes est que seul un sous-ensemble de colonnes est nécessaire pour obtenir la solution optimale d'un programme linéaire, puisque la plupart d'entre elles ne seront pas dans la base. La génération de colonnes démarre d'un ensemble arbitraire de couvertures initiales, soit un sous-ensemble

des colonnes de MP^{MNL}, afin que ce dernier admette au moins une solution réalisable. Le programme linéaire réduit est appelé problème maître restreint (RMP^{MNL}). À chaque itération, l'algorithme résout RMP^{MNL} pour obtenir la valeur optimale des variables duales, qui interviennent dans la fonction objectif de PP^{MNL}. Le rôle de PP^{MNL} est de trouver une couverture profitable, c'est-à-dire dont le coût réduit est strictement positif, ou de démontrer qu'il n'en existe pas. La résolution de PP^{MNL} donne une nouvelle couverture à ajouter à RMP^{MNL}. Itération après itération, l'ajout de couvertures de coût réduit positif améliore la valeur de la fonction objectif de RMP^{MNL}. Lorsqu'il n'est plus possible de trouver de couverture de coût réduit positif, l'optimalité de la solution courante de RMP^{MNL} est démontrée, et l'algorithme s'arrête. Une description plus détaillée de la génération de colonnes est donnée dans (Desrosiers *et al.* 2005 ; Lübbecke *et al.* 2005).

Variantes de MNL

PP^{MNL} est facilement adaptable, ce qui ouvre la voie à la résolution de nombreuses variantes. Dans (Rossi *et al.* 2010), la variante proposée impose des contraintes de bande passante, c'est-à-dire qu'une couverture ne peut activer plus de W capteurs. L'extension proposée dans (Rossi *et al.* 2011) tolère qu'une certaine proportion des cibles ne soit pas couverte à chaque instant. Le problème de Q-couverture présenté dans (Gu *et al.* 2007, 2009, 2011, 2012a,b) impose que chaque cible soit surveillée par un nombre minimum de capteurs à tout instant, afin de garantir une qualité de service en cas de défaillance des capteurs, au prix d'une moindre durée de vie du réseau. Ce problème est étendu au cas où la consommation énergétique dépend du nombre de cibles surveillées, et celui où les couvertures doivent assurer la connectivité avec une station de base (Singh *et al.* 2013b). Les capteurs ayant une portée ajustable peuvent modifier leur consommation énergétique en fonction de leur rayon de portée, qui devient une décision du problème. Les rayons de portée peuvent prendre leurs valeurs dans un ensemble discret (Cardei *et al.* 2005b ; Cerulli *et al.* 2012 ; Rossi *et al.* 2012) ou continu (Rossi *et al.* 2012). Dans (Singh *et al.* 2013a), les capteurs sont directionnels, et l'angle de cette zone est une décision supplémentaire du problème.

Il est aussi parfois nécessaire d'acheminer les données récoltées par les capteurs vers une station de base pour procéder à leur traitement, et cela peut être fait en transmettant les données d'un capteur à l'autre jusqu'à la station de base. Il convient donc de considérer un second réseau, dédié à la communication entre capteurs. Dans (Castaño *et al.* 2013 ; Raiconi *et al.* 2011), la connectivité des couvertures, c'est-à-dire l'existence d'un chemin

de communication disponible entre chaque capteur sélectionné et la station de base, est assurée. Cette variante est également étendue au cas où seulement une proportion des cibles nécessite d'être couverte (Castaño *et al.* 2014). Une autre variante prend en considération les coûts de communication dans la consommation énergétique des capteurs (Alfieri *et al.* 2007 ; Castaño *et al.* 2015 ; Zhao *et al.* 2008). L'utilisation de plusieurs stations de base, afin d'augmenter davantage la durée de vie du réseau, est proposée dans (Castaño *et al.* 2013). Kaddour (2015) organise les communications entre les capteurs, de manière à maximiser la durée de vie du réseau, en supposant que le temps est découpé en intervalles discrets.

Résolution du problème auxiliaire de MNL

Dans le cadre d'un algorithme de génération de colonnes, il n'est pas nécessaire de résoudre le problème auxiliaire à l'optimalité à chaque itération pour obtenir un ordonnancement optimal. Construire une couverture de coût réduit positif suffit à améliorer la solution de MNL. Par conséquent, l'utilisation de métaheuristiques pour résoudre le problème auxiliaire est tout à fait appropriée pour accélérer la génération de colonnes. Toutefois, pour garantir l'optimalité, il est indispensable de résoudre le problème auxiliaire de manière exacte au moins une fois, et en particulier à la dernière itération. La combinaison d'une métaheuristique et d'une méthode exacte à base de programmation linéaire en nombres entiers est appelée *matheuristique*. Dans cette section, nous offrons une vue d'ensemble des métaheuristiques utilisées dans la littérature.

La *Greedy Randomized Adaptive Search Procedure* (GRASP) (Feo *et al.* 1995) est une métaheuristique constructive facile à mettre en œuvre et offrant de bonnes performances sur le problème de couverture d'ensembles. Elle convient souvent à la résolution du problème auxiliaire, dont la structure est généralement similaire à celle du problème de couverture d'ensembles. Elle est paramétrée à l'aide d'un coefficient $\alpha \in [0, 1]$. Lorsque α est proche de 0, l'algorithme se comporte de façon aléatoire. Et inversement, lorsqu'il est proche de 1, il se comporte de façon déterministe. La nouvelle couverture est construite itérativement, en partant d'une couverture vide. À chaque itération, une utilité est attribuée à chaque capteur, en fonction de son énergie restante et du nombre de cibles non encore couvertes qu'il peut surveiller. Les capteurs ayant une utilité supérieure à un certain seuil, calculé en fonction de α, font partie d'une liste notée RCL (Restricted Candidate List). Un capteur est choisi aléatoirement dans la RCL pour être ajouté à la couverture, et une nouvelle itération commence, jusqu'à ce que toutes les cibles soient couvertes. GRASP est généralement

complétée par une recherche locale qui tente d'améliorer la couverture en effectuant des échanges de capteurs. Cette métaheuristique doit bien sûr être adaptée pour répondre aux contraintes des variantes de MNL, par exemple en ajoutant une procédure de réparation de solution. Cette métaheuristique est utilisée dans (Castaño *et al.* 2013 ; Castaño *et al.* 2013 ; Castaño *et al.* 2014 ; Raiconi *et al.* 2011).

Variable Neighborhood Search (VNS) (Hansen *et al.* 2009), est une méthode de recherche locale explorant successivement plusieurs voisinages. Elle part d'une solution initiale et la transforme en effectuant des modifications profitables (par exemple, la désactivation d'un capteur plus l'activation d'un autre, moins coûteux) sur celle-ci jusqu'à atteindre un optimum local. Ensuite, le voisinage est changé en choisissant un autre type de modification (par exemple, la désactivation de deux capteurs plus l'activation d'un autre). La recherche locale est ainsi relancée, pour chaque voisinage, jusqu'à ce que tous les voisinages aient été explorés ou qu'un autre critère d'arrêt soit vérifié. Cette métaheuristique est utilisée dans (Castaño *et al.* 2014), en complément de GRASP.

Les *algorithmes génétiques* (Goldberg 2006 ; Sastry *et al.* 2013) s'inspirent de la sélection naturelle et l'appliquent à un sous-ensemble de couvertures, appelé *population*, dans laquelle chaque *chromosome* représente une couverture. Ici nous décrivons brièvement une version élémentaire de cette méthode. À chaque itération (ou *génération*), deux chromosomes sont sélectionnés et appelés *parents*. Des opérations de *croisement* et de *mutation* sont appliquées successivement. Le croisement génère un nouveau chromosome (*enfant*) à partir des chromosomes parents. La mutation perturbe le chromosome enfant en le modifiant aléatoirement. Tous les chromosomes sont évalués par une fonction de *fitness* et la *sélection* est une étape où les chromosomes ayant obtenu le fitness le moins élevé sont supprimés de la population. Les algorithmes génétiques sont hautement adaptables et font l'objet de paramétrages (probabilité de croisement, de mutation, etc.). L'avantage principal des méthodes à population dans la résolution approchée des problèmes auxiliaires est qu'elles offrent la possibilité d'ajouter plusieurs couvertures profitables au problème maître réduit à chaque itération de la génération de colonnes, ce qui a généralement pour effet de faire diminuer sensiblement le nombre d'itérations permettant d'obtenir une solution optimale au problème maître. Ces méthodes sont utilisées dans (Carrabs *et al.* 2015a,b ; Rossi *et al.* 2011, 2012, 2013 ; Singh *et al.* 2013a,b).

Des heuristiques basées sur la relaxation linéaire du problème auxiliaire sont proposées dans Gu *et al.* (2011).

La résolution exacte du problème auxiliaire s'effectue généralement en dernier recours,

lorsque les métaheuristiques échouent ou pour prouver l'optimalité de la solution courante du problème maître réduit. Le plus souvent, la formulation par programmation linéaire en variables entières du problème auxiliaire est directement résolue par un solveur commercial tel que CPLEX (*IBM ILOG CPLEX Optimizer*). Toutefois, lorsque le problème auxiliaire est suffisamment complexe, il peut être approprié d'utiliser des techniques plus avancées. Castaño *et al.* (2015) proposent une méthode Branch-and-Cut basée sur la décomposition de Benders, qu'ils comparent à une approche à base de programmation par contraintes.

Autres problèmes

Dans ce paragraphe, nous énumérons d'autres problèmes étudiés, impliquant des cibles fixes.

Martins *et al.* (2007) et Pinheiro *et al.* (2013) minimisent la consommation d'énergie totale du réseau de capteurs, en tenant compte des coûts de communication, et proposent un programme linéaire en nombres entiers. Le problème présenté dans (Cheng *et al.* 2007) consiste à maximiser l'aire de la surface couverte par des capteurs directionnels, en décidant l'angle de ceux-ci. Le problème MCBB (Minimum Coverage under Bandwidth Constraints) (Cheng *et al.* 2005 ; Rossi *et al.* 2012 ; Wang *et al.* 2008) consiste à minimiser la non-couverture de cibles lorsque la durée de vie du réseau de capteurs est imposée. Cheng *et al.* (2005) et Wang *et al.* (2008) le résolvent à l'aide d'heuristiques basées sur la relaxation linéaire. Rossi *et al.* (2012) proposent une matheuristique à base de génération de colonnes couplée à un algorithme génétique.

Un problème plus général, dont les décisions incluent le placement de capteurs et de la station de base, ainsi que l'ordonnancement d'activités de surveillance, est étudié par Türkoğullari *et al.* (2007). Ce problème est formulé par un programme linéaire en variables mixtes, résolu de manière approchée à l'aide une méthode heuristique basée sur la relaxation Lagrangienne. Une extension comprenant le routage des données de communication est également formulée (Türkoğullari *et al.* 2010a,b). Elle est résolue par une méthode heuristique décomposée en deux phases. La première est un algorithme de génération de colonnes exécuté sur une relaxation du programme linéaire, et la deuxième est une heuristique exploitant les colonnes obtenues à la première phase (Türkoğullari *et al.* 2010a). Une autre méthode consiste à trouver des ensembles de capteurs connectés qui satisfont les contraintes de couverture, puis à décider des routes empruntées par les données jusqu'à la station de base (Türkoğullari *et al.* 2010b).

Des problèmes multi-objectifs ont été formulés (Jameii *et al.* 2015 ; Martins *et al.*

2011 ; Özdemir *et al.* 2012a,b ; Sengupta *et al.* 2012 ; Sengupta *et al.* 2013), dont les objectifs incluent la minimisation de l'énergie consommée ou la maximisation de l'aire de couverture, par exemple.

Algorithmes distribués

Les algorithmes distribués effectuent les calculs localement et conviennent donc particulièrement aux réseaux de capteurs sans fil, de part leur capacité à s'adapter à des réseaux de très grande taille. Lenzen *et al.* (2011) donnent une vue d'ensemble d'algorithmes distribués pour les réseaux de capteurs sans fil. Madan *et al.* (2006) proposent un algorithme de sous-gradient distribué, qui organise les flux de communication de manière à maximiser la durée de vie du réseau. Le problème est formulé par un programme linéaire en variables continues dont le lagrangien permet de formuler un sous-problème par capteur. Gatzianas *et al.* (2008) étudient une variante dans laquelle la station de base est mobile. D'une manière similaire, un programme linéaire est formulé dont la décomposition se prête bien à l'implémentation d'un algorithme de sous-gradient distribué. Zhao *et al.* (2012) proposent une méthode similaire pour une variante dont l'objectif est de maximiser l'utilité du réseau.

1.2.3 Surveillance de cibles mobiles

Dans de nombreux contextes, les cibles peuvent être des êtres vivants ou des objets qui se déplacent dans la zone de surveillance. Puisque les cibles sont mobiles, il n'est pas toujours nécessaire de couvrir toute la zone de surveillance, mais seulement les points où se situent les cibles, afin d'activer moins de capteurs. Cela nécessite de disposer de suffisamment d'informations sur le comportement des cibles. Dans le cas d'un train sur une voie ferrée, d'un véhicule routier sur une autoroute, ou d'un avion dans un couloir aérien, les trajectoires sont contraintes par les voies de circulation et leurs obstacles. Toutefois, sans connaissance parfaite du comportement des cibles, l'exploitant du réseau doit prendre en compte les incertitudes. L'une des idées les plus intuitives est de considérer une zone d'incertitude à couvrir, sous la forme d'un disque, dont le centre est la position supposée de la cible (Zhang *et al.* 2004). Le rayon peut être choisi de telle sorte que la cible ne peut quitter le disque d'incertitude avant l'acquisition de la position suivante. Cependant, les trajectoires des cibles sont parfois sujettes à des contraintes de mobilité dues aux caractéristiques physiques de la cible. Afin de réduire davantage le nombre de capteurs à activer, Jeong *et al.* (2007) proposent de réduire la zone d'incertitude en prenant en compte les lois de

la cinématique, en particulier pour le cas des véhicules terrestres. En effet, certaines zones du disque d'incertitude ne peuvent être visitées par un véhicule routier dans un temps suffisamment court, elles sont alors éliminées de la zone à couvrir.

Dans le domaine du suivi de cibles mobiles, la littérature abonde de protocoles de suivi et de communication, mais peu d'entre eux utilisent des techniques d'optimisation. L'état de l'art par Naderan *et al.* (2012) fait état d'une seule méthode, proposée par Lee *et al.* (2006), puis améliorée par Yeong-Sung *et al.* (2010), utilisant effectivement des techniques d'optimisation. À partir d'un programme linéaire en variables binaires, cette méthode calcule un arbre de communication grâce à une heuristique reposant sur la relaxation Lagrangienne. Les données d'entrée sont les fréquences des mouvements des cibles, en particulier les fréquences de passage de la zone de couverture d'un capteur à celle d'un autre, et son objectif est de minimiser la consommation énergétique liée aux communications.

Dans (Liu *et al.* 2009), l'objectif est de maximiser la durée de vie du réseau. Les capteurs peuvent avoir trois états : actif, au repos, éteint. Un capteur au repos est capable de communiquer avec les autres capteurs mais n'est pas en mesure de surveiller les cibles. La génération de couvertures est formulée comme un problème où l'on doit maximiser le nombre de capteurs éteints, en s'assurant qu'il existe des chemins de communication entre les capteurs actifs et une station de base. Les heuristiques proposées reposent sur des algorithmes de plus courts chemins.

Les protocoles applicables aux réseaux de capteurs sans fil sont généralement basés sur des topologies de communication particulières, comme des arbres ou des clusters, afin d'optimiser un ou plusieurs critères. Les topologies définissent la façon dont les capteurs communiquent entre eux et permettent d'organiser efficacement les flux de communications. Zhang *et al.* (2004) calculent successivement des arbres de coût minimum suivant les déplacements d'une cible, à l'aide d'un algorithme de programmation dynamique, de manière à trouver une topologie minimisant l'énergie consommée par les capteurs. Patel *et al.* (2005) proposent un programme linéaire en variables mixtes, pour organiser le réseau de capteurs en clusters. Le problème est décomposé afin d'appliquer un algorithme de génération de colonnes. Santos *et al.* (2012) organisent le réseau de capteurs en clusters, de manière à minimiser la consommation énergétique. Le problème est formulé comme un problème d'ensemble dominant indépendant, modélisé par un programme linéaire en nombres entiers et résolu à l'aide d'une métaheuristique basée sur GRASP.

1.2.4 Incertitude et robustesse

La robustesse est un aspect intéressant dans les problèmes de réseaux de capteurs sans fil, puisque dans la plupart des cas, le comportement des cibles est sujet à incertitude. De plus, le réseau de capteurs peut faire face à des attaques ou des dysfonctionnements. Dans ce cas, on considère la *survivabilité* du réseau de capteurs, c'est-à-dire sa capacité à remplir une mission donnée malgré une attaque ennemie ou la défaillance d'un ou plusieurs capteurs. Afin de pallier cette problématique, une solution est d'imposer que chaque cible soit surveillée par un minimum de capteurs afin de garantir sa couverture malgré la disparition d'un ou plusieurs capteurs. Ce problème est présenté typiquement comme le problème de Q-couverture, introduit dans la section 1.2.2 (page 8). Henna *et al.* (2013) suggèrent de générer des couvertures disjointes, afin que le dysfonctionnement d'un capteur n'impacte pas d'autres couvertures. Dans (Lin *et al.* 2011, 2012), une mesure appelée *Degree of Disconnectivity* (DoD) évalue les dommages causés par un attaquant sur un réseau, calculée en fonction du nombre de nœuds endommagés sur le plus court chemin entre chaque paire de nœuds source-destination. Afin de garantir les communications vers une station de base, Bari *et al.* (2012) formulent un programme linéaire minimisant le nombre de nœuds relais en garantissant que chaque capteur est couvert par au moins un nœud relais. Wang *et al.* (2013) proposent une formulation mathématique pour minimiser la probabilité de succès d'une attaque du réseau. La théorie des jeux permet de prendre les meilleures décisions en cas d'attaque ennemie sur le réseau de capteurs. Huangfu *et al.* (2014) étudient la relation entre la densité d'un réseau de capteurs et la survivabilité, dans un contexte où les capteurs tombent en panne aléatoirement. Un état de l'art (Shi *et al.* 2012) propose une vue d'ensemble des applications de la théorie des jeux dans les réseaux de capteurs sans fil. En supposant que la probabilité de défaillance de chaque capteur est connue, Tian *et al.* (2015) construisent des couvertures de cardinalité minimale, couvrant toutes les cibles avec une probabilité supérieure à un seuil donné. Le problème est formulé comme une variante du problème de couverture d'ensembles, un algorithme glouton construit un ordonnancement d'activités de capteurs.

Dans le contexte plus général de la recherche opérationnelle, l'*analyse de sensibilité* consiste à évaluer l'impact de la modification de paramètres d'un problème sur ses solutions. Il s'agit de réaliser l'étude de la stabilité d'une solution déjà calculée. Il existe une variété d'outils pour mesurer la robustesse (ou sensibilité) de la solution d'un problème (Gurevsky 2011 ; Kouvelis *et al.* 1997). L'une des plus connues est l'*intervalle de sensibilité* (Ward

et al. 1990). Supposons qu'une donnée initiale α de l'instance d'un problème soit sujette à des perturbations. Étant données une solution x et une propriété de x (par exemple son optimalité), l'intervalle de sensibilité de α est l'intervalle des valeurs de α pour lesquelles la propriété de x reste inchangée. Il s'agit par exemple, étant donné un ordonnancement d'activités de capteurs, de savoir dans quel intervalle de valeurs la capacité initiale d'un capteur peut varier tout en préservant l'admissibilité de l'ordonnancement.

Le *rayon de stabilité* (Sotskov *et al.* 1998, 2006) considère quant à lui un ensemble de paramètres. Étant donnés x une solution, une propriété de x, et $\mathcal{A}$ un ensemble de paramètres, le rayon de stabilité ρ d'une solution x est la plus petite variation des paramètres $\alpha \in \mathcal{A}$ pour laquelle x perd sa propriété. Autrement dit, chacun des paramètres $\alpha \in \mathcal{A}$ peut varier dans l'intervalle $[\alpha - \rho, \alpha + \rho]$ en garantissant que la solution x conserve sa propriété.

Lorsqu'une mesure de robustesse devient un objectif, alors on parle de problème d'optimisation robuste (Bertsimas *et al.* 2011). Ceci est le cas du problème présenté dans le chapitre 3, où l'on cherche à trouver une solution maximisant le rayon de stabilité. Kouvelis *et al.* (1997) présentent trois principaux critères de robustesse. L'un des plus connus est le *critère du pire cas* ou *robustesse absolue*. Ainsi, on cherche à calculer une solution qui soit optimale dans le pire des scénarios envisagés. Dans un scénario donné, le *regret* d'une solution correspond à l'écart entre la valeur de sa fonction objectif et la valeur optimale qu'elle aurait pu obtenir dans ce scénario. À partir de cette mesure, on peut minimiser le critère du *regret maximum* ou *minimax regret*, qui correspond au plus grand regret parmi tous les scénarios. Ce critère est approprié, par exemple, aux situations où l'on souhaite minimiser le manque à gagner. Le critère du *regret relatif* est similaire au regret absolu, à la différence qu'il considère l'écart relatif au lieu de l'écart absolu. Ces critères ont été étudiés pour un très grand nombre de problèmes, dont le problème de sac-à-dos, le problème du voyageur de commerce, ou les problèmes d'ordonnancement (Billaut *et al.* 2005 ; Gurevsky 2011).

1.2.5 Contributions

Les travaux présentés dans cette thèse introduisent des problèmes de suivi de cibles mobiles résolus avec les méthodes de la recherche opérationnelle. Les méthodes proposées pour résoudre ces problèmes permettent d'obtenir des solutions optimales en temps raisonnable pour les instances considérées.

Dans cette thèse, nous proposons tout d'abord une méthode préliminaire appelée *discrétisation*, dont l'objectif est de s'affranchir des données géométriques des problèmes.

Elle comprend deux variantes et est suffisamment générique pour convenir à tous les problèmes de suivi de cibles mobiles par des réseaux de capteurs sans fil. L'une des variantes est complétée par une procédure de réduction de la taille d'une instance, qui permet de réduire les temps de résolution dans de nombreux cas. La discrétisation sert de première phase aux méthodes présentées dans cette thèse.

Dans le problème présenté au chapitre 3, un réseau de capteurs surveille une cible sujette à incertitude temporelle. La trajectoire spatiale de la cible est connue, mais les temps de passage en chacun de ses points sont incertains. L'objectif consiste à construire un ordonnancement dit à énergie minimale (c'est-à-dire qu'un seul capteur est actif à tout instant) qui maximise le rayon de stabilité, c'est-à-dire un ordonnancement qui résiste au plus grand retard ou à la plus grande avance par rapport aux temps prévisionnels. L'algorithme proposé s'exécute en temps pseudo-polynomial. Il repose sur une dichotomie qui résout à chaque itération le problème de décision associé au problème d'optimisation. Le problème de décision étant formulé comme un problème de transport, il existe pour le résoudre des algorithmes s'exécutant en temps polynomial. La dichotomie s'exécute ainsi en un temps très raisonnable. Toutefois, des bornes supérieures très efficaces rencontrent souvent la valeur optimale et permettent de s'affranchir de la dichotomie dans de nombreux cas. L'algorithme proposé résout des instances comportant jusqu'à 1000 capteurs en moins de 15 secondes en moyenne.

Le problème présenté au chapitre 4 introduit la notion d'incertitude spatiale dans le cas du suivi de plusieurs cibles. Étant donnée une zone d'intérêt que les exploitants du réseau souhaitent pouvoir surveiller à long terme, l'objectif consiste à minimiser la consommation énergétique du réseau de capteurs pour accomplir la mission de surveillance courante, tout en maximisant la garantie de couverture de la zone d'intérêt. La garantie de couverture est la durée minimale, à l'issue de la mission courante, durant laquelle le réseau de capteurs est capable d'assurer la surveillance de tout point de la zone d'intérêt. Le problème est décomposé en trois sous-problèmes correspondant à trois étapes de la méthode proposée. Le premier sous-problème consiste à construire un ensemble de couvertures réalisable, le second maximise la garantie de couverture, et le troisième minimise la consommation d'énergie du réseau de capteurs une fois la garantie de couverture fixée. Les travaux présentent des programmes linéaires en variables continues et des bornes pour les différents sous-problèmes. La méthode proposée est une matheuristique basée sur la génération de colonnes, accélérée par la métaheuristique GRASP. Les trois sous-problèmes sont des problèmes maîtres formulés par des programmes linéaires, qui partagent le même ensemble de colonnes. Leur

particularité est de partager également le même problème auxiliaire, formulé comme une variante du problème de couverture d'ensembles.

Enfin, le chapitre 5 présente trois exemples d'extensions du problème MSR du chapitre 3. La première d'entre elles considère les coûts de communication des capteurs en prenant en charge l'acheminement des données de surveillance vers une station de base.

Dans la deuxième extension, le réseau de capteurs doit surveiller plusieurs cibles. Enfin, la troisième extension considère à la fois l'incertitude spatiale et temporelle pour la surveillance d'une cible. Pour chacune des extensions, nous avons esquissé une méthode afin d'ouvrir des perspectives de recherche.

1.2.6 Organisation du manuscrit

Le chapitre 1 présente un état de l'art et une vue d'ensemble de la littérature scientifique en relation avec cette thèse en optimisation dans les réseaux de capteurs sans fil.

Le chapitre 2 est dédié à la discrétisation, qui constitue la première phase des méthodes présentées dans les chapitres 3 et 4.

Le chapitre 3 présente un problème d'ordonnancement robuste d'activités de capteurs dans lequel une cible peut se présenter en avance ou en retard par rapport à une prévision donnée. L'objectif est de construire un ordonnancement de rayon de stabilité maximum, afin de maximiser son aptitude à rester admissible en dépit des avances et des retards de la cible, tout en respectant les contraintes de couverture et de capacité des batteries.

Le chapitre 4 présente un problème d'ordonnancement d'activités de capteurs dont l'objectif est de maximiser une garantie de couverture dans un contexte multi-cibles et multi-missions. Dans ce chapitre, les positions des cibles peuvent être sujettes à incertitude.

Enfin, le chapitre 5 conclut le manuscrit, propose des extensions et discute des perspectives de ces travaux.

Chapitre 2

Discrétisation

Contenu

Un réseau de capteurs est constitué d'une multitude de capteurs autonomes disséminés aléatoirement, par exemple largués depuis un avion ou un hélicoptère. Leur mission est de détecter ou surveiller des cibles dans une région où les infrastructures de surveillance sont inadaptées ou inexistantes, comme un champ de bataille ou une zone dévastée par une catastrophe naturelle. Le réseau de capteurs, ainsi que les cibles et leur comportement, peuvent être décrits à l'aide d'éléments géométriques. Ces éléments peuvent servir de données d'entrée à tous les problèmes étudiés dans cette thèse, indépendamment des contraintes ou de l'objectif visé, car ils constituent une description directe de l'environnement et du contexte. Dans un premier temps, nous donnons une modélisation géométrique du réseau de capteurs et des cibles. Ensuite, nous présentons la *discrétisation*, une refor-

mulation dont le but est de se ramener à un problème d'ordonnancement d'activités de capteurs. Cette procédure est un préalable indispensable à toutes les méthodes présentées dans cette thèse. Elle s'inscrit dans la lignée de méthodes utilisées pour les situations où les cibles sont immobiles. En effet, bien que les cibles soient mobiles, le temps peut être découpé en intervalles de telle sorte que, dans chaque intervalle, les cibles puissent être considérées immobiles sans altérer le problème. De ce fait, l'idée de la discrétisation s'inspire de travaux réalisés sur les cibles immobiles, où l'on réduit des domaines continus à des domaines discrets sans modifier le problème (Rossi *et al.* 2012, 2013).

2.1 Modélisation géométrique

Chaque capteur est capable de surveiller des cibles dans sa *zone de couverture*. À cause d'obstacles, d'irrégularités du terrain, ou d'interférences entre capteurs, la zone de couverture peut être de forme quelconque, comme sur la figure 2.1. Toutefois, sans perte de généralité, dans un espace en deux dimensions (respectivement trois dimensions), nous considérerons que ces zones sont des disques (des sphères) dont le centre est la position du capteur. La *zone de surveillance* est définie par l'union des zones de couverture.

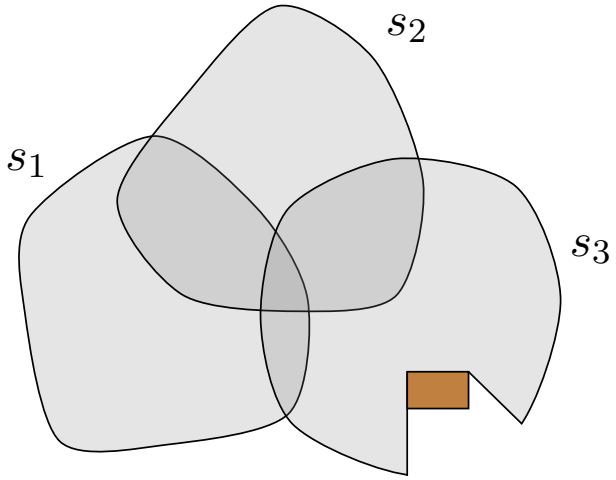


Figure 2.1 – Une zone de surveillance quelconque avec les zones de couverture de 3 capteurs, et un obstacle

Une cible à un instant t est représentée par un point. On suppose que la trajectoire d'une cible est donnée. Elle est modélisée par une fonction vectorielle continue $\mathcal{T} : \mathbb{R} \to \mathbb{R}^q$,

définie par $\mathcal{T}(t)$ où t est une variable de temps et $q \in \mathbb{N}^*$ est la dimension de l'espace.

Les notations utilisées dans la suite de cette thèse sont les suivantes. Le nombre de capteurs est noté m et le nombre de cibles n. Par convention, l'indice d'un capteur est noté i, celui d'une cible est noté j. L'ensemble des indices des capteurs est noté I, et l'ensemble des cibles est noté J. L'ensemble des points de la zone de couverture d'un capteur $i \in I$ est noté s_i. La trajectoire d'une cible $j \in J$ est notée $\mathcal{T}_j$. Les notations sont résumées dans le tableau 2.1.

I	Ensemble des capteurs $\{1, \ldots, m\}$
J	Ensemble des cibles $\{1, \ldots, n\}$
s_i	Zone de couverture du capteur i
$\mathcal{T}_j(t)$	Position de la cible j à l'instant t

TABLEAU 2.1 – Données d'entrée géométriques

Les éléments géométriques qui vont être présentés servent de données d'entrée à tous les problèmes traités dans cette thèse. Néanmoins, nous ne connaissons pas de technique de résolution exploitant directement ces informations, c'est-à-dire sans nécessiter de procédure de transformation préalable. C'est pour cette raison que les méthodes que nous présentons se déroulent en deux phases. La première phase, appelée *discrétisation*, est une étape commune à chacune d'entre elles. Le but de la discrétisation est de s'affranchir des notions géométriques. La seconde phase, spécifique à chaque problème, consiste à résoudre un problème d'ordonnancement dont les données sont produites par la procédure de discrétisation.

2.2 Partitionnement de la zone de surveillance

En deux dimensions, la zone de surveillance peut être vue comme un graphe planaire (BERMAN *et al.* 2004 ; SLIJEPCEVIC *et al.* 2001). Les nœuds sont les intersections entre les bords des zones de couverture. Les arêtes relient les nœuds le long de ces bords. Les surfaces bordées par les arêtes sont les faces. Tous les points à l'intérieur d'une face sont couverts par le même ensemble de capteurs. Si plusieurs cibles se trouvent dans une même face, alors un seul capteur suffit à les couvrir toutes. En trois dimensions, la zone de surveillance est partitionnée en volumes, bordés par des surfaces.

Lorsque les zones de couverture sont des disques, le nombre de faces ne peut pas dépasser $m(m-1)+2$ (BERMAN *et al.* 2004). Un cercle peut couper au plus deux fois tout autre

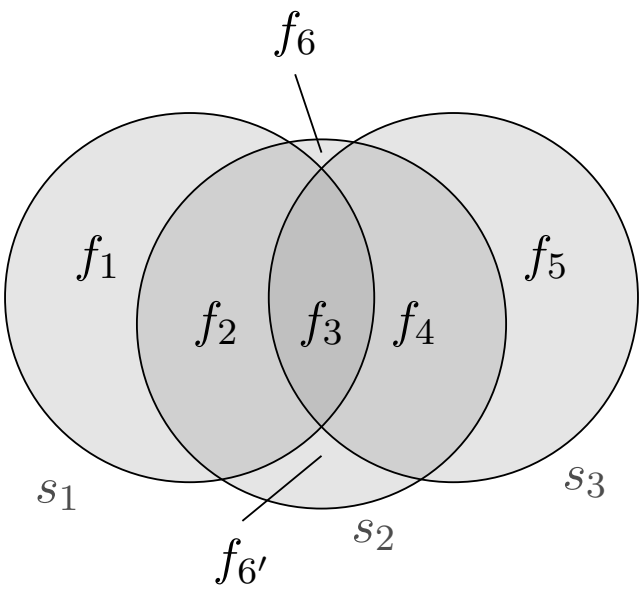


FIGURE 2.2 – Une zone de surveillance avec 3 capteurs, partitionnée en 6 faces

cercle. Si l'on suppose que chaque cercle coupe exactement deux fois chacun des autres cercles, alors le nombre de sommets $|V|$ est au maximum $m(m-1)$, et le nombre d'arêtes $|E|$ est au maximum $2m(m-1)$. Grâce à la formule d'Euler $|V| - |E| + |F| = 2$, où $|F|$ est le nombre de faces, $|F|$ ne peut dépasser $m(m-1)+2$, si on inclut la face extérieure non couverte.

Dans le cas en trois dimensions où les zones de couverture sont des sphères, le nombre de volumes ne peut pas dépasser $\frac{1}{3}m(m^2-3m+8)$ (GORDON *et al.* 1987), en incluant le volume extérieur. Pour des raisons de concision, le terme *face* sera à la fois utilisé pour les faces (cas en deux dimensions) et les volumes (cas en trois dimensions).

Dans la suite, une *face* est définie par un ensemble unique de capteurs. Plusieurs faces géométriques disjointes couvertes par exactement le même ensemble de capteurs seront considérées comme une seule et même face. Par exemple, sur la figure 2.2, les faces géométriques f_6 et $f_{6'}$ sont couvertes par l'ensemble $\{s_2\}$, elles sont alors définies comme une seule et même face. On note $\widehat{F}$ l'ensemble de toutes les faces, et $S(f) \subseteq I$ l'ensemble des capteurs couvrant la face $f \in \widehat{F}$.

2.3 Partitionnement de l'horizon de temps

Supposons que la mission de surveillance dure H unités de temps et qu'elle se déroule entre les dates $t = 0$ et $t = H$. Cet intervalle de temps est appelé *horizon de temps*. Chacune des cibles peut entrer ou sortir de la zone de surveillance à des instants différents. Certaines peuvent se présenter après $t = 0$, certaines peuvent disparaître avant $t = H$. On considérera, sans perte de généralité, que toute cible est présente dans la zone de surveillance pendant tout l'horizon de temps. En effet, une cible qui sort de la zone de surveillance peut être considérée comme couverte uniquement par un capteur fictif dont l'énergie est infinie, et qui ne couvre que les cibles se trouvant hors de la zone de couverture des capteurs de I.

Durant l'horizon de temps, toutes les cibles doivent être surveillées par au moins un capteur. Par conséquent, dans tous les problèmes traités dans cette thèse, il est nécessaire de s'assurer qu'à tout instant, un et un seul sous-ensemble de capteurs permettant de couvrir toutes les cibles est actif. On répond ainsi à la contrainte de couverture des cibles.

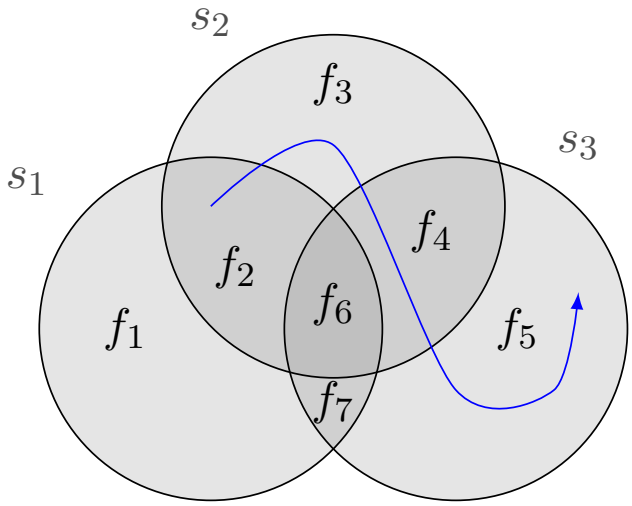


FIGURE 2.3 – Un réseau de 3 capteurs et la trajectoire d'une cible

La trajectoire d'une cible j peut être vue comme une séquence des faces traversées $(f_{\ell_1}, f_{\ell_2}, \ldots, f_{\ell_q})$ où ℓ_u est l'indice de la $u^{\text{ème}}$ face traversée. Par exemple, sur la figure 2.3, la cible traverse successivement les faces f_2, f_3, f_4 et f_5. Soit $F \subseteq \widehat{F}$ l'ensemble des faces traversées par les cibles. Tous les points d'une face $f \in F$ sont couverts par un même ensemble de capteurs candidats $S(f)$. Ainsi, pour couvrir une cible à un instant t, il est suffisant de connaître la face f dans laquelle elle se trouve et d'activer au moins un capteur

appartenant à l'ensemble $S(f)$. Les positions exactes des capteurs et des cibles ne sont alors plus nécessaires pour résoudre les différents problèmes de surveillance.

Les moments clés de la mission de surveillance sont les instants où une cible traverse la frontière d'une face. Lorsqu'une cible j change de face, l'ensemble des capteurs candidats pour sa surveillance change aussi. Ces dates de transition sont appelées *ticks*. Les ticks sont obtenus en calculant les instants où les cibles franchissent les frontières des faces, ce qui est possible dès lors que leurs trajectoires sont connues, en incluant le début et la fin de l'horizon de temps. Par définition, aucune cible ne change de face entre deux ticks consécutifs, et par conséquent les ensembles de capteurs candidats pour chaque cible demeurent inchangés. Ainsi, on peut voir chacun des problèmes de surveillance comme une séquence de problèmes de couverture d'ensembles.

Pour chaque cible j, on obtient une séquence $(t_1^j, t_2^j, \ldots, t_{n_j}^j)$ de ticks associés, où t_u^j est une date d'entrée ou de sortie de la zone de couverture d'un capteur, avec t_1^j et $t_{n_j}^j$ respectivement les dates d'entrée et de sortie de la zone de surveillance. Le tick t_u^j est la date d'entrée dans la $u^{\text{ème}}$ face, et t_1^j et $t_{n_j}^j$ sont respectivement les dates d'apparition et de disparition de la cible relativement à la zone de surveillance. Compte tenu de nos postulats de départ, nous considérons que $t_1^j = 0$ et $t_{n_j}^j = H$. Il peut arriver qu'une cible sorte d'une zone de couverture (s_i) et entre dans une autre $(s_{i'})$ au même instant. Dans ce cas, on considère deux ticks de même valeur. La face visitée entre ces deux ticks est une face qui n'est couverte ni par s_i, ni par $s_{i'}$.

Tous les problèmes traités dans cette thèse n'acceptent pas exactement les mêmes données d'entrée. Les ticks nécessitent notamment un traitement spécifique au problème à résoudre. Deux variantes de la discrétisation sont proposées. La première, appelée discrétisation *suivant les faces*, consiste à considérer que la couverture de cibles est équivalente à la couverture de faces. Cette variante convient à un problème où l'on optimise un critère énergétique. Une autre variante de la discrétisation est proposée, *suivant les cibles*. Elle convient au problème de robustesse présenté dans cette thèse.

2.4 Discrétisation suivant les faces

2.4.1 Introduction

Cette variante, qui est utilisée au chapitre 4, nécessite tout d'abord la fusion des séquences de ticks. Lorsque deux ticks ont la même valeur, un seul est conservé. Dans cette nouvelle séquence, les ticks sont notés t_k où k est la position du $k^{\text{ème}}$ tick. L'horizon

de temps est alors partitionné en p *fenêtres de temps*, où une fenêtre de temps est une période séparée par deux ticks consécutifs.

L'idée sous-jacente de cette variante est de considérer qu'une couverture de cibles est équivalente à une couverture de faces. En effet, activer un capteur pour couvrir une cible revient à couvrir tous les points de sa zone de couverture, et par conséquent, tous les points de la face dans laquelle la cible est située. Autrement dit, pour couvrir toutes les cibles, il est suffisant de couvrir les faces où elles sont situées. Pour chaque fenêtre de temps k, définissons un ensemble $T(k) \subseteq F$ des faces à couvrir. L'ensemble des cibles n'est alors plus nécessaire puisque les faces où elles se trouvent sont connues. Ainsi, un atout de la discrétisation *suivant les faces* est de s'affranchir de la notion de cible. Le résultat de cette discrétisation est résumé dans le tableau 2.2. On appelle ce résultat une *instance*, qui contient les données d'entrée d'un problème d'ordonnancement d'activités de surveillance.

I	Ensemble des capteurs $\{1, \ldots, m\}$
F	Ensemble des faces traversées $\{f_1, \ldots, f_q\}$
K	Ensemble des fenêtres de temps $\{1, \ldots, p\}$
$S(f) \subseteq I$	Ensemble des capteurs candidats couvrant la face $f \in F$
$T(k) \subseteq F$	Ensemble des faces à couvrir pendant la fenêtre de temps $k \in K$
t_k	Tick d'indice $k \in K \cup \{p+1\}$
Δ_k	Durée de la fenêtre de temps $k \in K$

TABLEAU 2.2 – Données résultant de la discrétisation suivant les faces

2.4.2 Réduction de la taille d'une instance

Dans cette section, la taille d'une instance est définie par la somme des cardinalités des ensembles de faces à couvrir, soit $\sum_{k \in K} |T(k)|$. Il est possible de réduire la taille de l'instance en supprimant des éléments des ensembles $T(k)$. Nous définissons une relation de dominance entre les faces.

Définition 1. (Relation de dominance entre les faces)
Soient une fenêtre de temps $k \in K$, et deux faces f et f' distinctes appartenant à $T(k)$. La face f domine f' si $S(f) \subset S(f')$.

Si une face $f \in T(k)$ domine $f' \in T(k)$, alors sélectionner un capteur de $S(f)$ suffit à couvrir à la fois f et f'. Par conséquent, il est possible de supprimer f' de l'ensemble $T(k)$. Supprimer toutes les faces dominées peut augmenter les chances d'obtenir des fenêtres de

temps k et k' telles que $T(k) = T(k')$. C'est une propriété désirable, puisqu'elle autorise la fusion de ces fenêtres de temps.

Algorithme 1 : Fusion des fenêtres de temps

```
// 𝒦 contient les indices triés des fenêtres de temps
∀k ∈ K, 𝒦(k) ← k
foreach k ∈ K do
    foreach k' ∈ K | k' ≥ k + 1 do
        if T(k) = T(k') then
            Δ_k ← Δ_k + Δ_k'
            K ← K \ {k'}
            𝒦(k') ← k   // Associe k' à k
```

L'algorithme 1 réduit le nombre de fenêtres de temps. S'il existe deux fenêtres de temps k et k' telles que $T(k) = T(k')$, alors ces fenêtres peuvent être fusionnées en une seule. La durée de la fenêtre de temps k' est ajoutée à celle d'indice k, et la fenêtre de temps k' est supprimée. L'indice k' est alors associé à k dans un vecteur d'indices noté $\mathcal{K}$. Par conséquent, la taille de l'instance est réduite et les fenêtres de temps originales peuvent être restaurées à l'aide du vecteur $\mathcal{K}$.

Pour réduire davantage la taille de l'instance, il est judicieux d'effectuer la suppression des faces dominées avant la fusion des fenêtres de temps, puisque la première augmente l'efficacité de la deuxième.

Lemme 1. *Une instance peut être réduite de telle sorte que le nombre de fenêtres de temps $|K|$ est inférieur ou égal à $2^{m(m-1)+1}$, où m est le nombre de capteurs.*

Démonstration. Soit $T(k)$ l'ensemble des faces à surveiller durant la fenêtre de temps k. Si $T(k) = T(k')$ avec $k \neq k'$, alors les fenêtres de temps k et k' peuvent être fusionnées Ainsi la fenêtre de temps k' est supprimée et la durée de la fenêtre de temps k est étendue à $\Delta_k + \Delta_{k'}$. Cette procédure d'élimination est répétée tant qu'il existe deux fenêtres de temps ayant le même ensemble de faces. Puisque le nombre de faces est inférieur ou égal à $m(m-1)+2$ (Berman *et al.* 2004), alors le nombre maximal de fenêtres de temps est inférieur ou égal à $2^{m(m-1)+1}$. Cette borne supérieure peut être améliorée en considérant $F_{max} = \max_{k \in K} |T(k)|$, le nombre maximal de faces à couvrir dans une fenêtre de temps. Par conséquent, le nombre de fenêtres de temps après réduction est borné par $2^{F_{max}-1} \leq 2^{m(m-1)+1}$. □

Ce lemme montre qu'après réduction, le nombre maximum de fenêtres de temps est indépendant du nombre de cibles et de leurs trajectoires. Néanmoins, en pratique, peu de fenêtres de temps sont produites lorsque les cibles visitent peu de faces.

2.5 Discrétisation suivant les cibles

Le problème abordé au chapitre 3 ne permet pas de s'affranchir de la notion de cible, ce qui justifie l'introduction de la discrétisation suivant les cibles. À l'instar de la discrétisation suivant les faces, on effectue une fusion des séquences de ticks. Cependant, deux ticks de même valeur ne sont pas nécessairement fusionnés. Les ticks sont partitionnés en deux catégories : les ticks *entrants* et les ticks *sortants*. Les ticks entrants correspondent à la date d'entrée d'une cible dans la zone de couverture d'un capteur, tandis que les ticks sortants correspondent aux instants où une cible quitte la zone de couverture d'un capteur. Par convention, le premier et le dernier tick sont respectivement sortant et entrant. Si deux ticks de même catégorie ont la même valeur, alors ces ticks sont fusionnés. Lorsqu'une cible entre dans une zone de couverture et sort d'une autre simultanément, alors ces ticks sont considérés comme étant distincts. Autrement dit, un tick entrant et un tick sortant ne sont pas fusionnés lorsqu'ils sont de même valeur.

Dans la nouvelle séquence, les ticks sont notés t_k où k est la position du $k^{\text{ème}}$ tick. À chaque tick, on attribue une valeur σ_k qui est définie en fonction du type de tick. À un tick t_k entrant, on attribue $\sigma_k = 1$ et à un tick sortant $t_{k'}$, $\sigma_{k'} = -1$. Soit l'ensemble $S_j(k)$, qui définit l'ensemble des capteurs candidats à la couverture de la cible j pendant la fenêtre de temps k. Le résultat de la discrétisation suivant les cibles est résumé dans le tableau 2.3.

I	Ensemble des capteurs $\{1, \ldots, m\}$
J	Ensemble des cibles $\{1, \ldots, n\}$
K	Ensemble des fenêtres de temps $\{1, \ldots, p\}$
$S_j(k) \subseteq I$	Ensemble des capteurs candidats couvrant la cible $j \in J$ pendant la fenêtre de temps $k \in K$
t_k	Tick d'indice $k \in K \cup \{p+1\}$
σ_k	Valeur déterminant la catégorie du tick k (entrant ou sortant)
Δ_k	Durée de la fenêtre de temps $k \in K$

TABLEAU 2.3 – Données résultant de la discrétisation suivant les cibles

Cette variante de la discrétisation est appropriée pour les méthodes nécessitant des opérations sur les ticks et les capteurs candidats en fonction des cibles. La méthode proposée

dans le chapitre 3 génère des instances modifiées dont les ticks de l'instance originale ont été déplacés. Les ensembles de capteurs candidats y sont également soumis à des modifications.

2.6 Exemple

2.6.1 Données d'entrée

Considérons un exemple simple en deux dimensions. En pratique, il est commode de représenter la trajectoire d'une cible à l'aide d'une ligne polygonale, c'est-à-dire une suite de segments dont la seconde extrémité de chacun d'entre eux est la première du suivant. Il est possible d'atteindre une précision aussi bonne que l'on veut puisque les trajectoires sont continues. De plus, lorsque les zones de couverture sont des disques, cette représentation réduit la discrétisation à une résolution d'équations linéaires du second degré à coefficients constants.

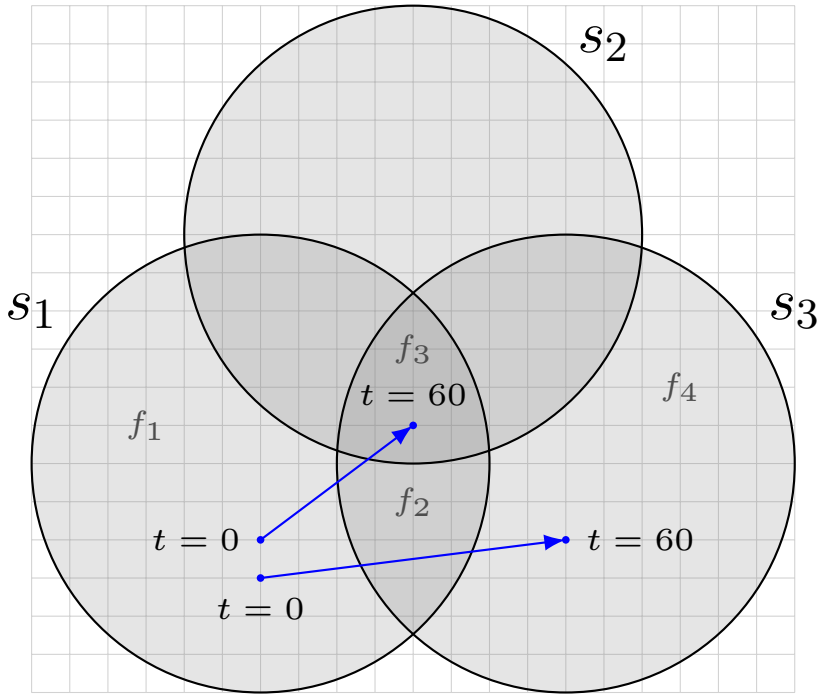


FIGURE 2.4 – Exemple d'un réseau de 3 capteurs et 2 cibles

La figure 2.4 représente un réseau de trois capteurs. Les zones de couverture s_1, s_2 et s_3 sont des disques de centres respectifs $(6, 6)$, $(10, 12)$ et $(14, 6)$ et de rayon 6. Deux cibles

se déplacent dans la zone de surveillance. Leur trajectoire est un segment et leur vitesse est uniforme. Le segment de la première cible relie le point de coordonnées $(6, 4)$ à celui de coordonnées $(10, 7)$. Celui de la deuxième cible relie les points $(6, 3)$ et $(13, 4)$. La durée de parcours de chacune des cibles est $H = 60$ unités de temps.

En calculant les intersections entre les trajectoires et les bords des zones de couverture, on obtient une séquence de ticks pour chaque cible :

$$\text{Ticks pour la cible 1 : } 0,\ 30.2948,\ 42.3344,\ 60$$
$$\text{Ticks pour la cible 2 : } 0,\ 22.3372,\ 47.8375,\ 60$$

Dans la suite, les deux variantes de la discrétisation sont illustrées.

2.6.2 Discrétisation suivant les faces

L'union des séquences de ticks donne le vecteur t contenant 6 ticks, donc 5 fenêtres de temps.

$$t = (\ 0,\ 22.3372,\ 30.2948,\ 42.3344,\ 47.8375,\ 60\ )$$

Les cibles visitent au total 5 faces différentes, indicées de 1 à 5. Les capteurs candidats couvrant ces faces sont stockés dans des ensembles $S(f)$ où f est l'indice de la face. Les faces à couvrir durant chaque fenêtre de temps k sont stockées dans des ensembles $T(k)$.

$$\begin{array}{lll} S(1) = \{s_1\} & T(1) = \{f_1\} & \Delta_1 = 22.3372 \\ S(2) = \{s_1, s_3\} & T(2) = \{f_1, f_2\} & \Delta_2 = 7.9576 \\ S(3) = \{s_1, s_2, s_3\} & T(3) = \{f_2\} & \Delta_3 = 12.0395 \\ S(4) = \{s_3\} & T(4) = \{f_2, f_3\} & \Delta_4 = 5.50317 \\ & T(5) = \{f_3, f_4\} & \Delta_5 = 12.1625 \end{array}$$

Dans certaines fenêtres de temps, le nombre de faces à couvrir peut être réduit sans altérer les solutions du problème. Tout d'abord, on observe que couvrir la face f_1 revient aussi à couvrir la face f_2, puisque $S(1) \subset S(2)$. Par conséquent, il n'est pas nécessaire de conserver la face f_2 de l'ensemble $T(2)$. Par un raisonnement analogue, on en déduit que

f_3 peut être supprimé de l'ensemble $T(4)$ car $S(2) \subset S(3)$, et f_3 peut être supprimé de $T(5)$ puisque $S(4) \subset S(3)$.

$$\begin{aligned} T(1) &= \{f_1\} & \Delta_1 &= 22.3372 \\ T(2) &= \{f_1\} & \Delta_2 &= 7.9576 \\ T(3) &= \{f_2\} & \Delta_3 &= 12.0395 \\ T(4) &= \{f_2\} & \Delta_4 &= 5.50317 \\ T(5) &= \{f_4\} & \Delta_5 &= 12.1625 \end{aligned}$$

Ensuite, la procédure de réduction suggère de fusionner les fenêtres de temps dans lesquelles les faces à couvrir sont identiques. On initialise un vecteur $\mathcal{K} = (1, 2, 3, 4, 5)$ qui contient les indices des fenêtres de temps. Durant les fenêtres de temps 1 et 2, les faces à couvrir sont identiques. La durée de la deuxième fenêtre de temps est ajoutée à celle la première et la deuxième est supprimée. Afin de restaurer les fenêtres de temps originales plus tard, on sauvegarde l'indice de la fenêtre de temps supprimée en affectant $\mathcal{K}(2) = 1$. Après réindexation des fenêtres de temps, on obtient une instance avec seulement 3 fenêtres de temps.

$$\begin{aligned} T(1) &= \{f_1\} & \Delta_1 &= 30.2948 \\ T(2) &= \{f_2\} & \Delta_2 &= 17.5427 \\ T(3) &= \{f_4\} & \Delta_3 &= 12.1625 \end{aligned}$$

$$\mathcal{K} = (1, 1, 2, 2, 3)$$

2.6.3 Discrétisation suivant les cibles

L'union des séquences de ticks donne le vecteur t contenant 6 ticks, donc 5 fenêtres de temps. Les ticks entrants sont marqués par le symbole "+" et les ticks sortants par le symbole "−".

$$t = (\ 0^-,\ 22.3372^+,\ 7.9576^+,\ 12.0395^+,\ 56.0374^-,\ 60^+\ )$$

Pour chacune des cibles j, on construit les ensembles $S_j(k)$ de capteurs candidats pour la fenêtre de temps k. Chaque valeur σ_k est égale à 1 si le tick t_k est entrant, -1 si le tick t_k est sortant.

$S_1(1) = \{s_1\}$	$S_2(1) = \{s_1\}$	$\Delta_1 = 22.3372$	$\sigma_1 = -1$
$S_1(2) = \{s_1\}$	$S_2(2) = \{s_1, s_3\}$	$\Delta_2 = 7.9576$	$\sigma_2 = 1$
$S_1(3) = \{s_1, s_3\}$	$S_2(3) = \{s_1, s_3\}$	$\Delta_3 = 12.0395$	$\sigma_3 = 1$
$S_1(4) = \{s_1, s_2, s_3\}$	$S_2(4) = \{s_1, s_3\}$	$\Delta_4 = 5.50317$	$\sigma_4 = 1$
$S_1(5) = \{s_1, s_2, s_3\}$	$S_2(5) = \{s_3\}$	$\Delta_5 = 12.1625$	$\sigma_5 = -1$
			$\sigma_6 = 1$

Chapitre 3

Ordonnancement robuste sous incertitude

Contenu

3.1 Introduction

En l'absence de données fiables sur le comportement des cibles, la prise en compte de l'incertitude s'impose dans les missions de surveillance. Dans ce chapitre, nous traitons le cas de la surveillance d'une cible unique. Le chemin emprunté par la cible est connu et une prévision des temps de passage en chacun de ses points est donnée. Ceci peut être le cas d'un train sur une voie ferrée ou un véhicule terrestre se déplaçant sur une route. Le

but est de planifier, avant la mission, un ordonnancement d'activités de capteurs à énergie minimale, capable d'assurer la surveillance malgré les aléas, qui peuvent affecter la cible, et qui se traduisent par une avance ou un retard par rapport aux temps de passage prévus.

3.2 Description du problème

Rappelons brièvement le contexte. Un ensemble de m capteurs $I = \{1, \ldots, m\}$ est déployé aléatoirement dans une région. Sans perte de généralité, nous considérons que la cible évolue sur une surface plane, et que les données géométriques du problème sont en deux dimensions. Tout capteur $i \in I$, lorsqu'il est actif, est capable de surveiller des cibles dans sa zone de couverture s_i, définie par un disque de rayon R^S. La zone de surveillance du réseau de capteurs considéré est l'union des zones de couverture des capteurs qui le constituent. Chaque capteur $i \in I$ est équipé d'une batterie de capacité E_i, exprimée en unités de temps. L'action de surveiller des cibles est appelée *activité* de surveillance, et consomme de l'énergie sur la batterie. Lorsqu'un capteur est inactif, la capacité de sa batterie demeure inchangée.

On suppose sans perte de généralité que la cible se déplace à l'intérieur de la zone de surveillance. En effet, si la cible quitte momentanément la zone de surveillance du réseau, alors il n'est plus possible de la surveiller, et on peut toujours supposer qu'elle est couverte par un capteur fictif d'énergie infinie, qui n'est plus disponible quand la cible se trouve dans la zone de surveillance. Nous disposons d'une prévision de sa trajectoire, exprimée à l'aide d'une fonction vectorielle continue $\mathcal{T} : \mathbb{R} \to \mathbb{R}^2$, définie par $\mathcal{T}(t)$ où t est une variable de temps. Sans perte de généralité, le domaine de t considéré est l'intervalle $[0, H]$ où H détermine la durée prévisionnelle de la mission. La cible se déplace le long de la trajectoire spatiale sans possibilité d'en sortir. En revanche, les dates de passage en chacun de ses points sont incertaines, c'est-à-dire que la cible peut être en retard ou en avance par rapport à la prévision. Cette situation est typique dans le cas d'un train sur une voie ferrée ou d'un avion dans un couloir aérien. Les données d'entrée du problème traité dans ce chapitre sont résumées dans le tableau 3.1.

L'objectif visé est de planifier la surveillance de la cible en protégeant la solution contre l'avance ou le retard par rapport aux temps de passage prévisionnels, c'est-à-dire en garantissant la surveillance permanente de la cible pendant toute la durée de la mission. On souhaite concevoir l'ordonnancement d'activités de surveillance le plus robuste possible, sans surcoût énergétique. Afin de respecter cette dernière contrainte, un seul capteur peut

être activé à chaque instant.

Pour rappel, la résolution des problèmes présentés dans cette thèse se déroule en deux phases. La première est la discrétisation, dont le but est de formuler un problème d'ordonnancement d'activités de capteurs, qui est résolu dans la deuxième phase.

I	Ensemble des capteurs $\{1, \ldots, m\}$
s_i	Zone de couverture du capteur i
E_i	Capacité du capteur i
$\mathcal{T}(t)$	Position de la cible à l'instant t

TABLEAU 3.1 – Données d'entrée géométriques

3.2.1 Discrétisation

La discrétisation est une reformulation des données d'entrée s'affranchissant des notions géométriques, détaillée dans le chapitre 2. Dans cette section, nous résumons brièvement les principes de la discrétisation. La zone de surveillance est partitionnée en un ensemble $\widehat{F}$ de faces disjointes, qui sont des surfaces couvertes par un même ensemble de capteurs (BERMAN *et al.* 2004 ; MEGUERDICHIAN *et al.* 2003 ; SLIJEPCEVIC *et al.* 2001). La position exacte de la cible n'est alors plus nécessaire puisque sa trajectoire peut être représentée par une séquence des faces traversées. Les dates de passage d'une face à une autre sont appelées *ticks* et les intervalles de temps séparant deux ticks consécutifs sont des *fenêtres de temps*. L'ensemble des fenêtres de temps est noté K, leurs durées respectives Δ_k, et les ticks sont notés t_k. On note p le nombre de fenêtres de temps. La variante utilisée dans ce chapitre est la discrétisation suivant les cibles, comme décrit dans la section 2.5. Le tableau 3.2 résume le résultat de la discrétisation. Puisqu'il n'y a qu'une seule cible à surveiller et pour des raisons de clarté, les ensembles de capteurs candidats $S_1(k)$ pour chaque fenêtre de temps k seront notés $S(k)$.

K	Ensemble des fenêtres de temps $\{1, \ldots, p\}$
$S(k) \subseteq I$	Ensemble des capteurs candidats couvrant la cible pendant la fenêtre de temps $k \in K \cup \{0, p+1\}$
t_k	Tick d'indice $k \in K \cup \{p+1\}$
σ_k	Valeur déterminant la catégorie du tick k (entrant ou sortant)
Δ_k	Durée de la fenêtre de temps $k \in K$

TABLEAU 3.2 – Données résultant de la discrétisation suivant les cibles

Pour rappel, les ticks sont partitionnés en deux catégories : les ticks *entrants* et les ticks *sortants*. Lorsque la cible entre dans une zone de couverture, le tick t_k correspondant est entrant. Dans ce cas, $S(k-1) \subset S(k)$. Lorsqu'elle sort d'une zone de couverture, le tick $t_{k'}$ correspondant est sortant, auquel cas $S(k') \subset S(k'-1)$. Par convention, le premier et le dernier tick sont respectivement sortant et entrant. Si deux ticks de même catégorie ont la même valeur, alors un seul est conservé. Lorsqu'une cible entre dans une zone de couverture et sort d'une autre simultanément, alors ces ticks sont considérés comme distincts même s'ils sont associés à la même date. Autrement dit, un tick entrant et un tick sortant ne sont pas fusionnés lorsqu'ils sont de même valeur. Dans ce cas particulier, l'ensemble des capteurs candidats entre ces deux ticks notés t_k et t_{k+1} est $S(k-1) \cap S(k+1)$.

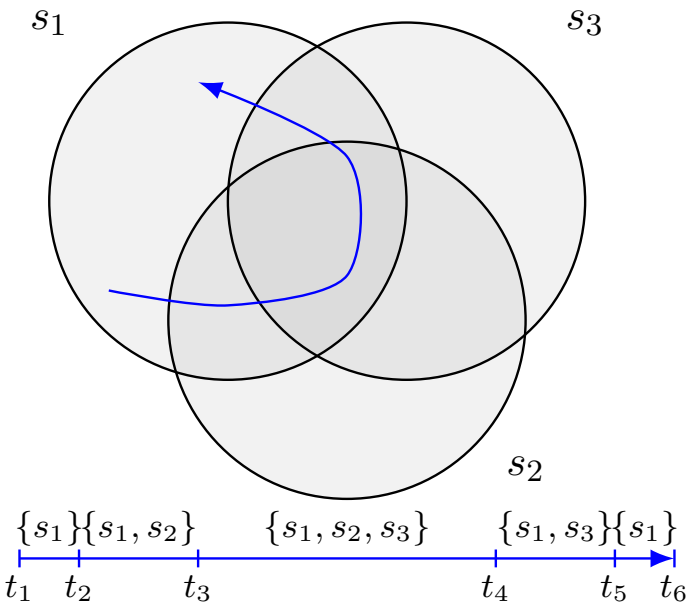


FIGURE 3.1 – Capteurs candidats le long d'une trajectoire

La figure 3.1 illustre un exemple avec 3 capteurs. En bas de la figure est représentée une frise chronologique montrant les capteurs candidats le long de la trajectoire. Par convention, l'ensemble des capteurs candidats avant t_1 et après t_{p+1} (sur la figure, t_6) est I, l'ensemble de tous les capteurs, donc $S(0) = S(p+1) = I$. Avec ces données, il est possible de calculer les *intervalles de disponibilité* des capteurs, illustrés par la figure 3.2.

Un intervalle de disponibilité d'un capteur i est un intervalle de temps durant lequel il est candidat. Même si le cas ne se présente pas dans l'exemple de la figure 3.1, un capteur

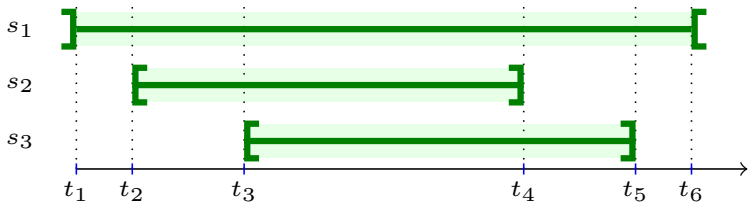


FIGURE 3.2 – Intervalles de disponibilité des capteurs

peut avoir plusieurs intervalles de disponibilité, ceux-ci étant alors deux à deux disjoints. Un capteur est disponible à un instant t si la cible se trouve dans sa zone de couverture. Les intervalles de disponibilité commencent par un tick entrant et terminent par un tick sortant, à l'exception de ceux qui commencent par t_1 ou se terminent par t_{p+1}. Puisqu'on ne peut pas activer plus d'un capteur à la fois, il n'est pas possible d'activer un capteur en dehors de ses intervalles de disponibilité, auquel cas ce capteur empêcherait l'activation d'un capteur couvrant la cible. Les intervalles de disponibilité sont utiles au calcul d'une mesure de robustesse, appelée *rayon de stabilité*, d'un ordonnancement d'activités de surveillance.

3.2.2 Rayon de stabilité

Les données résultant de la discrétisation sont les données d'entrée d'un problème d'ordonnancement d'activités de surveillance appelé MSR, pour *Maximization of the Stability Radius*. Pour rappel, la cible ne peut sortir de sa trajectoire spatiale. Néanmoins, les dates de passage en chacun des points de sa trajectoire sont incertaines. Un tick t_k est la date de passage prévue à la position $\mathcal{T}(t_k)$. Autrement dit, à l'instant t_k, la cible est attendue à la frontière entre deux faces, notées f_{k-1} et f_k. La face f_{k-1} est couverte par les capteurs de l'ensemble $S(k-1)$ et f_k est couverte par les capteurs de $S(k)$. Si la cible arrive à la frontière avec un retard de ρ unités de temps, alors elle quitte la face f_{k-1} à l'instant $t_k + \rho$. Par conséquent, afin d'éviter la perte de couverture dans ce scénario, un capteur de l'ensemble $S(k-1) \cap S(k)$ devrait être activé pendant l'intervalle $[t_k, t_k + \rho]$. Par symétrie, si la cible arrive avec une avance de ρ unités de temps, alors elle devrait être surveillée par un capteur de l'ensemble $S(k-1) \cap S(k)$ pendant l'intervalle $[t_k - \rho, t_k]$. Afin de supporter à la fois le retard et l'avance de la cible, un des capteurs parmi $S(k-1) \cap S(k)$ doit être activé durant l'intervalle $[t_k - \rho, t_k + \rho]$.

La cible peut également arriver en avance au point $\mathcal{T}(t_1)$, lieu d'apparition de la cible. Si la cible y apparaît à l'instant $t_1 - \rho$, alors l'ordonnancement devrait commencer à cette

même date. Par symétrie, l'ordonnancement devrait se terminer après la date t_{p+1}, afin d'anticiper un éventuel retard en $\mathcal{T}(t_{p+1})$.

Les avances ou retards de la cible sont susceptibles de faire s'écarter les ticks de leur valeur prévisionnelle. Soit $\mathcal{S}$ un ordonnancement. L'objectif du rayon de stabilité est de mesurer la plus grande variation des ticks t_k qui ne compromet pas la capacité de l'ordonnancement $\mathcal{S}$ à couvrir continuellement la cible sans la perdre de vue. La notion de rayon de stabilité est introduite dans les références (Sotskov *et al.* 1998, 2006). Un ordonnancement dont le rayon de stabilité est égal à ρ est capable de couvrir la cible, qu'elle subisse un retard ou une avance de ρ unités de temps.

En particulier, le rayon de stabilité est défini par la durée minimale séparant un instant où un capteur est actif et un instant où il n'est plus candidat. Autrement dit, le rayon de stabilité est limité par la durée entre les frontières d'une activité et les frontières de l'intervalle de disponibilité de ses capteurs candidats.

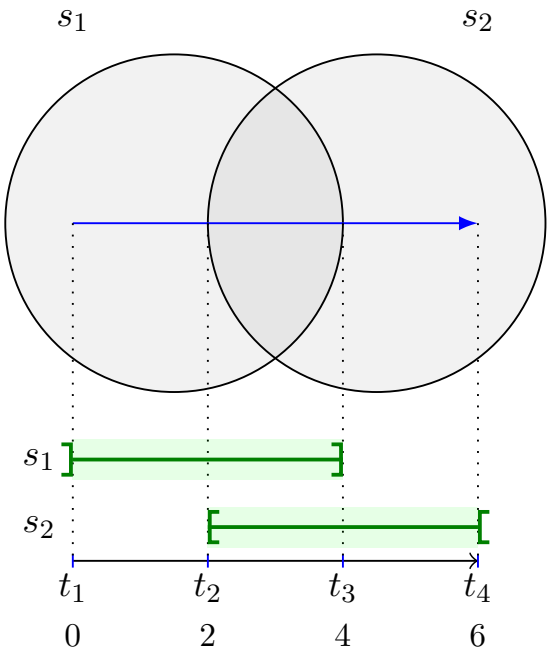
La figure 3.3a illustre un exemple simple avec deux capteurs. On suppose que la capacité des batteries est infinie. L'ordonnancement $\mathcal{S}_1$ (figure 3.3b) active le capteur 1 durant l'intervalle $[0, 4]$ et le capteur 2 durant l'intervalle $[4, 6]$. Tandis que l'ordonnancement $\mathcal{S}_2$ (figure 3.3c) active le capteur 1 pendant l'intervalle $[-1, 3]$ et le capteur 2 pendant l'intervalle $[3, 7]$. Les deux ordonnancements sont réalisables puisque les capacités sont infinies et la cible est couverte à chaque instant dans le cas où les temps de passage prévisionnels de la cible ne sont pas perturbés.

Supposons que la cible apparaisse une unité de temps plus tôt que prévu, et qu'elle disparaisse une unité de temps plus tard que prévu. On observe que l'ordonnancement $\mathcal{S}_1$ n'est pas pas réalisable pour un tel scénario, puisqu'il commence à $t = 0$ et se termine à $t = H$. De plus, la cible arrive 1 unité de temps plus tôt à la position $\mathcal{T}(t_3)$. Puisque le capteur 1 ne couvre pas la face après t_3, $\mathcal{S}_1$ n'est pas capable de surveiller la cible dans ce scénario, son rayon de stabilité est nul. En revanche, l'ordonnancement $\mathcal{S}_2$ résiste à toutes ces perturbations et son rayon de stabilité a pour valeur 1.

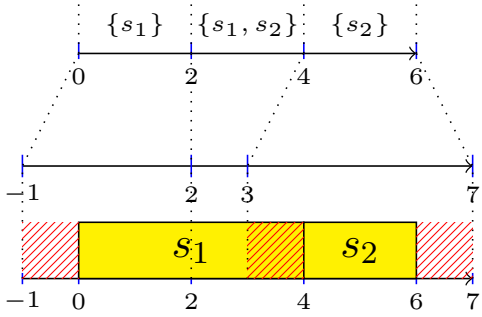
Dans ce chapitre, notre but est de créer un ordonnancement d'activités dont le rayon de stabilité est maximal.

3.2.3 Bornes supérieures sur le rayon de stabilité

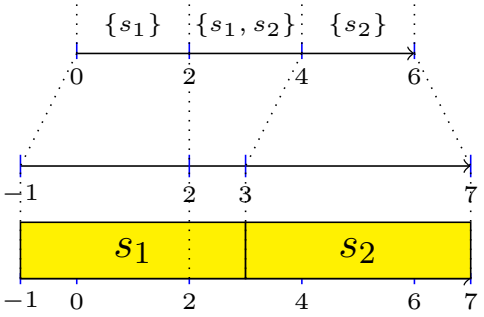
Nous proposons deux types de bornes supérieures sur le rayon de stabilité. Toutes les bornes présentées reposent sur l'idée d'étendre la durée des fenêtres de temps jusqu'à épuisement des capteurs. Le premier type de borne est basé sur les durées entre les fenêtres



(a) Exemple avec 2 capteurs



(b) Exemple d'ordonnancement non-robuste $\mathcal{S}_1$



(c) Exemple d'ordonnancement robuste $\mathcal{S}_2$

FIGURE 3.3 – Exemples d'ordonnancements robustes

de temps et les capteurs candidats qu'elles ont en commun. Le second type repose sur les capacités des capteurs candidats.

Bornes basées sur les durées entre les fenêtres de temps

La borne notée UB1 est obtenue en calculant la plus petite durée entre deux fenêtres de temps n'ayant aucun capteur candidat en commun.

$$\text{UB1} = \min_{k,k' \in K} \left\{ \frac{1}{2} \left(t_{k'} - t_{k+1} \right) \mid k < k' \text{ et } S(k) \cap S(k') = \emptyset \right\}$$

S'il existe deux fenêtres de temps k et k' telles que $S(k)$ et $S(k')$ sont disjoints, alors il n'existe aucun capteur capable de couvrir simultanément les deux faces correspondant à ces fenêtres de temps. Or, l'une des contraintes du problème est qu'on ne peut activer qu'un seul capteur à la fois. À chaque instant, on active soit un capteur parmi $S(k)$, soit un capteur parmi $S(k')$, et seulement une des faces peut être couverte. Par conséquent, l'extension de la durée des deux fenêtres de temps est limitée par la moitié de la durée qui les sépare.

Cette borne supérieure peut être généralisée pour chaque paire de fenêtres de temps. Lorsque deux fenêtres de temps partagent des capteurs candidats en commun, le rayon de stabilité est limité par la somme des capacités des batteries des capteurs communs à ces fenêtres. La généralisation de UB1 est notée UB1′.

$$\text{UB1}' = \min_{k,k' \in K} \left\{ \frac{1}{2} \left(\sum_{i \in S(k) \cap S(k')} E_i + t_{k'} - t_{k+1} \right) \mid k < k' \right\}$$

Ainsi, dans le cas où deux fenêtres de temps k et k' ne partagent aucun capteur candidat, alors $\sum_{i \in S(k) \cap S(k')} E_i = 0$ et UB1′ est réduite à UB1.

Bornes basées sur les capacités des capteurs

À chaque fenêtre de temps est associée une face, et donc un ensemble de capteurs. La durée de séjour additionnelle de la cible à l'intérieur d'une face ne peut excéder la somme des capacités des capteurs candidats de cette face. UB2 utilise ces capacités pour fournir une borne supérieure du rayon de stabilité.

$$\text{UB2} = \min_{k \in K} \left\{ \frac{1}{2} \left(\sum_{i \in S(k)} E_i - (t_{k+1} - t_k) \right) \right\}$$

La généralisation de cette borne repose sur le principe qu'une face donnée peut être associée à différentes fenêtres de temps. Ceci se produit lorsque la cible traverse plusieurs fois une même face. Soit F l'ensemble des faces traversées. À chaque face $f \in F$ est associé S_f, l'ensemble des capteurs couvrant celle-ci, et ρ_f une borne supérieure sur le rayon de stabilité calculée à partir de cette face. Initialement, ρ_f est nul. Tout d'abord, on calcule l'ensemble $K_f = \{k \in K | S(k) \subseteq S_f\}$ des fenêtres de temps pour lesquelles les capteurs candidats sont un sous-ensemble de S_f. $\mathcal{D}$ est la liste contenant les durées entre chaque paire de fenêtres de temps consécutives de K_f. Les $|K_f| - 1$ valeurs de $\mathcal{D}$ sont supposées triées par ordre croissant. $|K_f|$ fenêtres de temps doivent être couvertes par des capteurs de S_f. Cela signifie que le rayon de stabilité est limité par la durée d'activation maximale des capteurs de S_f après couverture des fenêtres de K_f, divisée par $2|K_f|$.

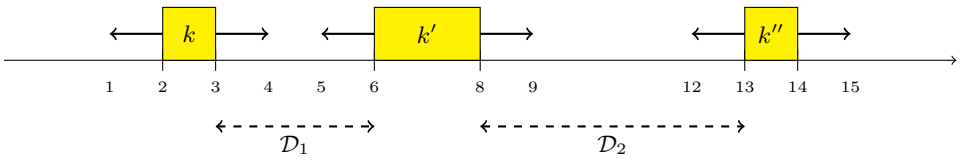


FIGURE 3.4 – Illustration de UB2′, où $\rho_f = 1$ et $\sum_{i \in S(f)} E_i = 10$

Considérons une face f visitée à trois reprises par la cible. K_f contient les fenêtres de temps k, k' et k'', montrées dans la figure 3.4. On suppose que deux capteurs de zones de couvertures s_1 et s_2 couvrent la face f, avec respectivement $E_1 = 5$ et $E_2 = 5$ unités de temps de capacité de batterie. Sur la figure 3.4, on peut déduire que la couverture des 3 fenêtres de temps requiert $1 + 2 + 1 = 4$ unités de temps. La capacité résiduelle des deux capteurs totalise alors $E_1 + E_2 - 4 = 6$. Par conséquent, puisque $|K_f| = 3$, le rayon de stabilité est inférieur ou égal à $\rho_f = \frac{6}{2|K_f|} = 1$.

Néanmoins, la figure 3.4 montre que si l'énergie totalisée par les capteurs de $S(f)$ est plus élevée, le rayon de stabilité peut atteindre 1.5. Dans ce cas, les fenêtres de temps k et k' sont dorénavant adjacentes, et il n'est plus possible de les étendre dans 6 directions mais seulement 4. Le cas où $\sum_{i \in I} E_i = 15$ est illustré sur la figure 3.5.

Enfin, si $E_1 + E_2$ est une valeur suffisamment grande, alors ρ_f atteint une valeur supérieure à 2.5, et les fenêtres de temps k et k' deviennent adjacentes. L'extension ne

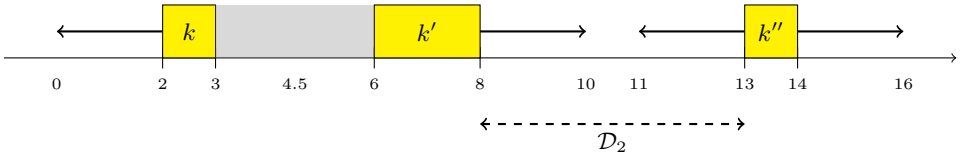


FIGURE 3.5 – Illustration de UB2′, où $\rho_f = 2$ et $\sum_{i \in S(f)} E_i = 15$

FIGURE 3.6 – Illustration de UB2′, où $\rho_f = 3$ et $\sum_{i \in S(f)} E_i = 18$

peut alors plus se faire que selon deux directions (figure 3.6).

Par conséquent, le calcul de ρ_f est effectué pas-à-pas. Soient $k_1 < k_2 < \cdots < k_{|K_f|}$ les fenêtres de temps de K_f. Pour chaque $q \in \{1, \ldots, |K_f| - 1\}$, on définit $\mathcal{D}_{k_q} = t_{k_q+1} - t_{k_q}$ la durée entre deux fenêtres de temps consécutives de K_f. Ces $|K_f| - 1$ durées sont réindicées par ordre croissant ($\mathcal{D}_1 \leq \cdots \leq \mathcal{D}_{|K_f|-1}$).

Algorithme 2 : Calcul de UB2′

$\rho_{\min} \leftarrow \infty$
foreach $f \in \hat{F}$ **do**
  $r \leftarrow \sum_{i \in S_f} E_i - \sum_{k \in K_f} (t_{k+1} - t_k)$
  while $|\mathcal{D}| > 0$ *and* $\frac{r}{2(|\mathcal{D}|+1)} > \frac{\min \mathcal{D}}{2}$ **do**
    $r \leftarrow r - \min \mathcal{D}$
    $\mathcal{D} \leftarrow \mathcal{D} \setminus \{\min \mathcal{D}\}$
  $\rho_{\min} \leftarrow \min \left\{ \rho_{\min}, \frac{r}{2(|\mathcal{D}|+1)} \right\}$
return $\rho_{\min}$

L'algorithme 2 retourne la plus petite borne supérieure calculée sur le rayon de stabilité. Il calcule, pour chaque face f, la capacité résiduelle r totalisée après surveillance des fenêtres de temps dans K_f. Tant que la capacité résiduelle est suffisamment grande, les fenêtres de temps les plus proches sont fusionnées et la durée correspondante $\min \mathcal{D}$ est supprimée de la liste $\mathcal{D}$. Le nombre $2(|\mathcal{D}| + 1)$ de directions d'extension est alors diminué. L'algorithme s'arrête lorsque l'extension maximale autorisée à chaque direction, $\frac{r}{2(|\mathcal{D}|+1)}$ est inférieure

ou égale à la moitié de la durée $\min \mathcal{D}$ entre les fenêtres de temps les plus proches.

Lemme 2. *UB1' et UB2' ne donnent aucune garantie de performance sur le rayon de stabilité ρ.*

Démonstration. On peut construire une instance dont le rayon de stabilité maximal est $\rho = 0$, avec $\text{UB1}' > 0$ et $\text{UB2}' > 0$, illustrée dans la figure 3.7. Cette instance est composée de 3 capteurs numérotés de 1 à 3, respectivement disponibles durant les intervalles $[0, 5]$, $[2, 10]$ et $[7, 12]$. La capacité initiale de chaque capteur i est $E_i = 4$. Il existe un ordonnancement réalisable pour $\rho = 0$, qui active le capteur 1 durant $[0, 4]$, le capteur 2 durant $[4, 8]$ et le capteur 3 durant $[8, 12]$. La somme des capacités des capteurs est 12, égale à l'horizon de temps. Par conséquent, le rayon de stabilité, nul, ne peut pas être augmenté, et l'ordonnancement proposé est optimal.

La borne supérieure $\text{UB1}'$ est limitée par la durée séparant les fenêtres de temps 1 et 3, ainsi que 3 et 5, ne partageant aucun capteur en commun. On obtient $\text{UB1}' = 1.5$.

La face à couvrir durant la fenêtre de temps 2 est couverte par les capteurs 1 et 2. La somme de leurs capacités est égale à 8. Puisque les ensembles des capteurs candidats des fenêtres 1 et 3 sont inclus dans l'ensemble des capteurs de la fenêtre de temps 2, les capteurs 1 et 2 seront activés au moins 7 unités de temps. Par conséquent, il reste une unité de temps pour étendre les fenêtres de temps, dans deux directions. On obtient la plus petite borne $\text{UB2}' = 0.5$. □

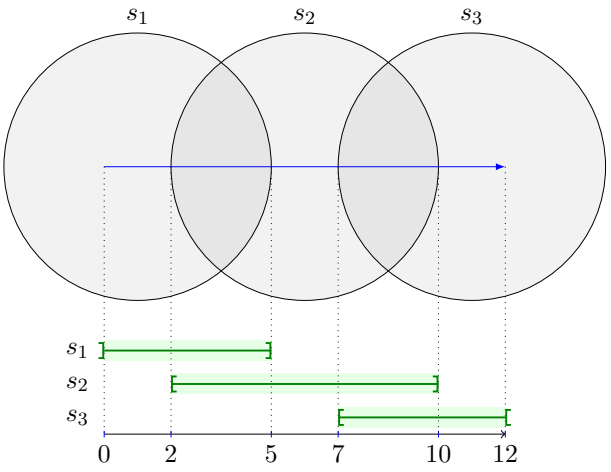


FIGURE 3.7 – Instance pour laquelle $\rho = 0$ et $\min\{\text{UB1}', \text{UB2}'\} > 0$

3.3 Résolution de MSR

Dans cette section, nous introduisons le problème de décision associé au problème d'ordonnancement. Ensuite, nous décrivons la méthode de résolution, exploitant le problème de décision.

3.3.1 Problème de décision

Soit ρ une valeur arbitraire positive ou nulle. Le problème de décision consiste à déterminer s'il existe un ordonnancement réalisable d'activités de surveillance tel que son rayon de stabilité est supérieur ou égal à ρ. Ce problème peut être formulé comme un problème de transport, qui est résolu à l'aide d'un algorithme basé sur l'algorithme de Kuhn pour les problèmes d'affectation (Ford *et al.* 1956).

Dans un premier temps, nous considérons le cas où $\rho = 0$. Une instance $\mathcal{I}_0$ du problème de transport est construite en attribuant aux capteurs le rôle de fournisseurs (de capacité E_i) et aux fenêtres de temps le rôle de clients (demande Δ_k). Les variables de décision sont appelées x_{ik} et désignent la durée allouée au capteur i pour surveiller la cible durant la fenêtre de temps k. Pour obtenir un problème de transport équilibré, la somme des demandes doit être égale à la somme des capacités. On crée une nouvelle demande $\Delta_{p+1} = \sum_{i\in I} E_i - \sum_{k\in K} \Delta_k$, et les variables x_{ip+1} déterminent les capacités résiduelles des capteurs. Les coûts de transport c_{ik} ont le rôle de pénalités afin d'empêcher l'activation de capteurs non-candidats. Ainsi,

$$\forall i \in I, k \in K : c_{ik} = \begin{cases} 0 & \text{si } k = p+1 \text{ ou } i \in S(k) \\ 1 & \text{sinon} \end{cases}$$

L'objectif est de minimiser la durée d'activation de capteurs non-candidats. Le problème de transport est formulé à l'aide d'un programme linéaire :

$$\min \sum_{i\in I} \sum_{k\in K\cup\{p+1\}} c_{ik} x_{ik} \tag{3.1}$$

$$\sum_{k\in K\cup\{p+1\}} x_{ik} = E_i \qquad \forall i \in I \tag{3.2}$$

$$\sum_{i\in I} x_{ik} = \Delta_k \qquad \forall k \in K \cup \{p+1\} \tag{3.3}$$

$$x_{ik} \geq 0 \qquad \forall i \in I, k \in K \cup \{p+1\} \tag{3.4}$$

La contrainte (3.2) assure que la durée totale d'activité d'un capteur n'excède pas la durée de vie de sa batterie. La contrainte (3.3) impose que la cible soit surveillée pendant toute la durée des fenêtres de temps.

Il existe un ordonnancement réalisable d'activités de surveillance si et seulement si la valeur optimale de la fonction objectif du problème de transport défini par (3.1)-(3.4) est nulle. Il est possible de construire un tel ordonnancement à partir d'une solution du problème de transport. Les variables x_{ik} peuvent être vues comme un budget de temps d'activité. Pour chaque valeur de x_{ik} strictement positive, on planifie l'activation du capteur i pendant la fenêtre de temps k, durant x_{ik} unités de temps. En s'assurant qu'elles ne se chevauchent pas, les activités sont séquencées dans un ordre arbitraire dans chaque fenêtre de temps.

Afin de vérifier l'existence d'un ordonnancement réalisable pour une valeur de $\rho > 0$, une nouvelle instance $\mathcal{I}_\rho$ est générée. L'instance $\mathcal{I}_\rho$ doit être créée de telle sorte que s'il existe un ordonnancement réalisable pour $\mathcal{I}_\rho$, alors cet ordonnancement a un rayon de stabilité d'au moins ρ pour $\mathcal{I}_0$. Puisque le rayon de stabilité est limité par les durées entre les activités et les bornes de leurs intervalles de disponibilité correspondants, une telle instance peut être obtenue en diminuant les intervalles de disponibilité pour interdire les activités susceptibles de réduire le rayon de stabilité à une valeur inférieure à ρ.

Dans la section 3.2.1, nous avons introduit deux catégories de ticks : les ticks entrants et les ticks sortants. Pour construire $\mathcal{I}_\rho$, les ticks entrants de $\mathcal{I}_0$ sont retardés de ρ unités de temps, tandis que les ticks sortants sont avancés de ρ unités de temps. Dès lors qu'un tick entrant et un tick sortant se rencontrent, leur position est échangée dans la séquence des ticks, et l'ensemble des capteurs candidats $S(k)$ est mis à jour.

$$S(k) \leftarrow S(k-1) \cap S(k+1)$$

La procédure est résumée dans l'algorithme 3. S'il existe une fenêtre de temps $k \in K$ telle que $S(k) = \emptyset$ et $\Delta_k > 0$, alors l'instance courante est non-réalisable.

La figure 3.8a montre un exemple d'instance $\mathcal{I}_0$ avec trois capteurs, où la trajectoire de la cible est décrite par un segment et son sens est de la gauche vers la droite. L'instance $\mathcal{I}_{1.5}$ peut être construite en deux étapes. Tout d'abord, les ticks entrants et sortants sont respectivement déplacés de $+1$ et -1 unités de temps (figure 3.8b). On obtient l'instance $\mathcal{I}_1$ dans laquelle les ticks t_3 et t_4 se rencontrent en un même instant. Par conséquent, ils sont échangés, et le capteur 3 n'est plus candidat entre ces deux ticks. Sur la figure 3.8b,

Algorithme 3 : Construction de l'instance I_ρ

Input : $\mathcal{I}_0 = (\{t_k\}, \{S(k)\})$ l'instance originale, ρ le rayon de stabilité

$\forall k \in K, \quad \sigma_k \leftarrow \begin{cases} -1 & \text{si } t_k \text{ est un tick sortant} \\ 1 & \text{si } t_k \text{ est un tick entrant} \end{cases}$

$\forall k \in K \cup \{p+1\}, \quad t'_k \leftarrow t_k$

$\forall k \in K, \quad S'(k) \leftarrow S(k)$

$\delta \leftarrow \rho$

while $\delta > 0$ **do**

  // Demi-distance minimale entre deux ticks opposés

  $\Delta_{\min} \leftarrow \min \left\{ \delta, \ \min_{k \in K} \frac{t'_{k+1} - t'_k}{2} \mid \sigma_{k+1} - \sigma_k < 0 \right\}$

  foreach $k \in K \cup \{p+1\}$ **do**

    $t'_k \leftarrow t'_k + \sigma \times \Delta_{\min}$ // Décale tous les ticks

  foreach $k \in K$ **do**

    // Échange les ticks sortants et entrants qui se rencontrent

    if $\sigma_{k+1} - \sigma_k < 0$ ***and*** $t'_{k+1} - t'_k \leq 0$ **then**

      $\sigma_k \leftarrow -\sigma_k$

      $\sigma_{k+1} \leftarrow -\sigma_{k+1}$

      $S'(k) \leftarrow S'(k-1) \cap S'(k+1)$

  $\delta \leftarrow \delta - \Delta_{\min}$

// $\mathcal{K}$ contient les indices triés des fenêtres de temps

$\forall k \in K, \quad \mathcal{K}(k) \leftarrow k$

foreach $k \in K$ **do**

  foreach $k' \in K | k' \geq k+1$ **do**

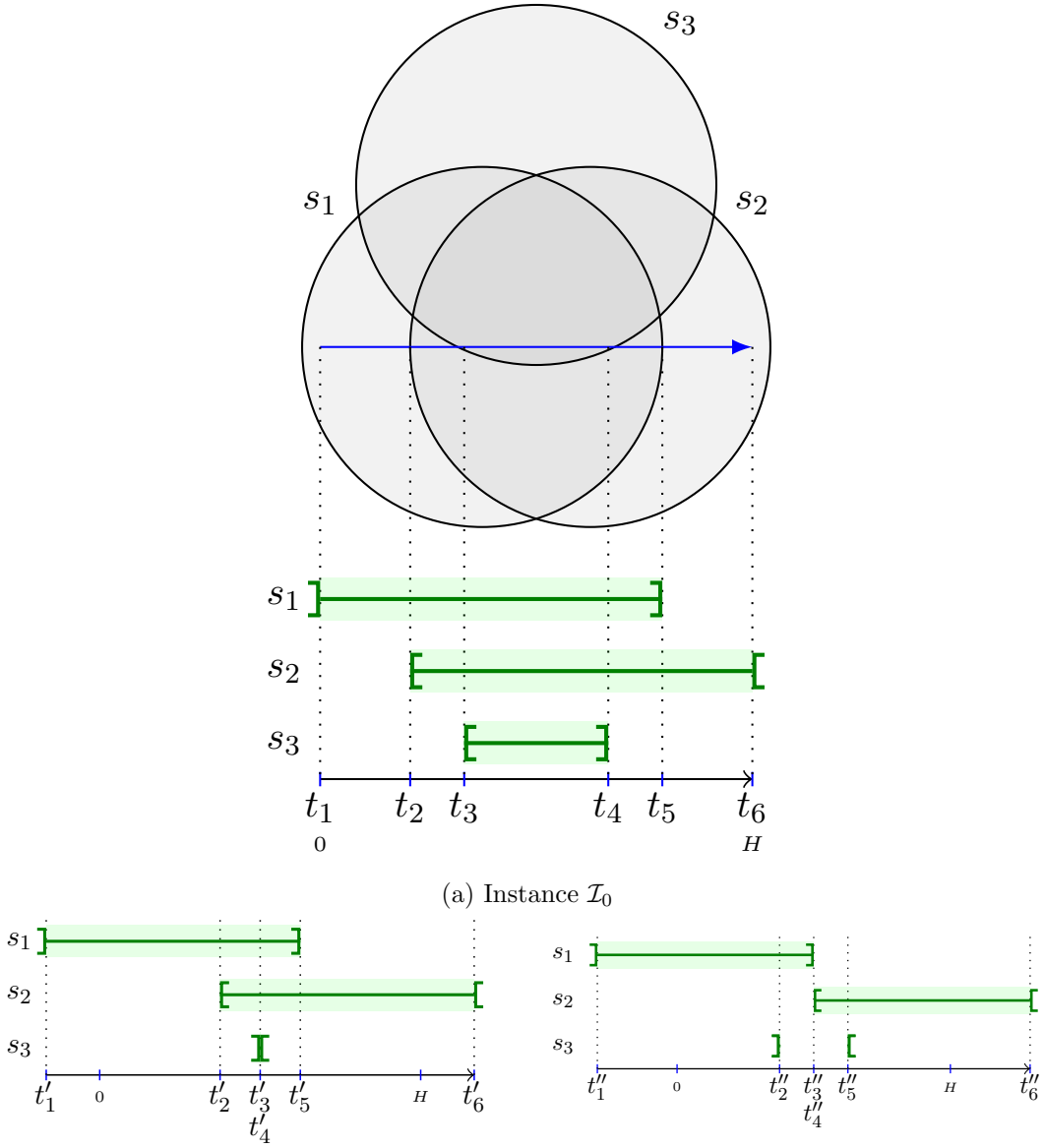
    if $T(k) = T(k')$ **then**

      $\Delta_k \leftarrow \Delta_k + \Delta_{k'}$

      $K \leftarrow K \setminus \{k'\}$

      $\mathcal{K}(k') \leftarrow k$ // Associe k' à k

return $\mathcal{I}_\rho = (\{t'_k\}, \{S'(k)\})$ // Nouvelle instance $\mathcal{I}_\rho$ avec t'_k et $S'(k)$



(a) Instance $\mathcal{I}_0$

(b) Instance $\mathcal{I}_1$

(c) Instance $\mathcal{I}_{1.5}$

FIGURE 3.8 – Construction d'instances $\mathcal{I}_\rho$

ceci est montré par l'intervalle nul séparé par des crochets ouverts sur la ligne du capteur 3. La même procédure est ensuite appliquée en déplaçant les ticks de 0.5. Enfin, l'instance $\mathcal{I}_{1.5}$ de la figure 3.8c garantit un rayon de stabilité d'au moins 1.5.

3.3.2 Problème d'ordonnancement

Le problème d'ordonnancement consiste à construire un ordonnancement qui maximise le rayon de stabilité ρ. L'algorithme 4 résout ce problème à l'aide d'une dichotomie sur ρ. Tout d'abord, il teste l'existence d'un ordonnancement réalisable pour $\rho = 0$ et $\rho = \text{UB}$, où $\text{UB} = \min\{\text{UB1}', \text{UB2}'\}$. S'il existe un ordonnancement réalisable pour $\rho = \text{UB}$, alors UB est la valeur du rayon de stabilité et le problème est résolu. S'il n'existe pas d'ordonnancement réalisable pour $\rho = 0$, cela signifie que le problème est non-réalisable. Dans le cas contraire, l'algorithme crée une liste ordonnée $\mathcal{D}$ contenant toutes les distances positives $t_{k'} - t_k < UB$ où t_k est un tick entrant et $t_{k'}$ un tick sortant. Les éléments de $\mathcal{D}$ sont triés par ordre croissant. L'algorithme trouve la valeur maximale $\mathcal{D}_\ell$ pour laquelle il existe un ordonnancement réalisable de rayon de stabilité $\rho \geq \mathcal{D}_\ell$ par dichotomie. À chaque itération de celle-ci, le problème de décision est résolu sous la forme d'un problème de transport (fonction SolveTP). Les bornes sur ρ, c'est-à-dire $\mathcal{D}_{\ell_{\min}}$ et $\mathcal{D}_{\ell_{\max}}$, sont mises à jour en fonction de la valeur optimale de la fonction objectif du problème de transport.

Lorsque la dichotomie est terminée, puisque $\rho \in [\mathcal{D}_{\ell_{\min}}, \mathcal{D}_{\ell_{\max}}[$, un programme linéaire est résolu pour obtenir la valeur finale, et donc optimale, de ρ. Les variables du programme linéaire sont les suivantes :

— $\delta \geq 0$: augmentation de ρ par rapport à $\mathcal{D}_{\ell_{\min}}$
— $x_{ik} \geq 0$: durée allouée au capteur $i \in I$ pour la surveillance dans la fenêtre de temps $k \in K$

L'objectif est de maximiser l'augmentation de la valeur de ρ. Le programme linéaire est le suivant :

$$\max \delta \tag{3.5}$$

$$\sum_{k \in K | i \in S(k)} x_{ik} \leq E_i \qquad \forall i \in I \tag{3.6}$$

$$\sum_{i \in S(k)} x_{ik} = \Delta_k + (\sigma_{k+1} - \sigma_k)\delta \qquad \forall k \in K \tag{3.7}$$

$$\delta \geq 0 \tag{3.8}$$

$$x_{ik} \geq 0 \qquad \forall k \in K, i \in S(k) \tag{3.9}$$

$$\text{où } \sigma_k = \begin{cases} 1 & \text{si } t_k \text{ est un tick entrant} \\ -1 & \text{si } t_k \text{ est un tick sortant} \end{cases}$$

La contrainte (3.6) assure que la durée d'activation d'un capteur n'excède pas la capacité de sa batterie. La contrainte (3.7) alloue le temps d'activation nécessaire pour assurer la surveillance de la cible dans chaque fenêtre de temps.

Algorithme 4 : Calcul du rayon de stabilité

$UB \leftarrow$ `ComputeUpperBound()`
if $UB < 0$ **then** Return -1 // Problème non-réalisable
// Existe-t-il un ordonnancement tel que $\rho = UB$?
$z \leftarrow$ `SolveTP(`$\mathcal{I}_{UB}$`)`
if $z = 0$ **then return** $\rho = UB$ // UB est le rayon de stabilité
// Existe-t-il un ordonnancement réalisable ?
$z \leftarrow$ `SolveTP(`$\mathcal{I}_0$`)`
if $z > 0$ **then return** -1 // Problème non-réalisable
// Dichotomie sur $\mathcal{D}$ liste ordonnée
$\mathcal{D} \leftarrow \{t_{k'} - t_k \in [0, UB[\mid t_k \text{ est entrant et } t_{k'} \text{ est sortant}\}$
$\ell_{\min} \leftarrow 1$
$\ell_{\max} \leftarrow |\mathcal{D}|$
while $\ell_{\max} - \ell_{\min} > 0$ **do**
  $\ell \leftarrow \left\lfloor \frac{\ell_{\min} + \ell_{\max}}{2} \right\rfloor$
  $\rho \leftarrow \mathcal{D}_\ell$
  $z \leftarrow$ `SolveTP(`$\mathcal{I}_\rho$`)`
  if $z = 0$ **then**
    $\ell_{\min} \leftarrow \ell$
  else
    $\ell_{\max} \leftarrow \ell$
$\rho \leftarrow \mathcal{D}_{\ell_{\min}}$
// Valeur finale de ρ
$\delta \leftarrow$ `SolveLP(`$\mathcal{I}_\rho$`)`
$\rho \leftarrow \rho + \delta$
return ρ

Lorsque la solution optimale de ce programme linéaire est obtenue, la valeur du rayon de stabilité est incrémentée de δ, et l'ordonnancement est construit en appliquant la méthode décrite dans la section 3.3.1. Dans l'ordonnancement obtenu, la durée d'activation totale

des capteurs est $H+2\rho$, puisqu'il débute ρ unités de temps avant t_1 et se termine ρ unités de temps après t_{p+1}. Les trois lemmes qui suivent déterminent la finitude de ρ et la complexité du problème traité dans ce chapitre.

Lemme 3. *ρ est fini si et seulement si $\bigcap_{k\in K} S(k) = \emptyset$ ou $\forall i \in \bigcap_{k\in K} S(k)$, E_i est finie.*

Démonstration. Tout d'abord, si $\bigcap_{k\in K} S(k) = \emptyset$, alors la cible traverse au moins deux faces ne partageant aucun capteur candidat en commun. Par conséquent, il n'existe aucun ensemble de capteurs capable de couvrir toute la trajectoire et ρ doit être finie. Si pour tout i appartenant à $\cap_{k\in K} S(k)$, E_i est finie, alors le rayon de stabilité est au plus $\text{UB1}' \leq \frac{1}{2}\left(\sum_{i\in\cap_{k\in K} S(k)} E_i + H\right)$. Inversement, si ρ est fini, alors tout capteur i capable de couvrir toute la trajectoire, s'il existe, a une capacité E_i finie. □

Lemme 4. *MSR peut être résolu en un nombre pseudo-polynomial d'itérations, en $\mathcal{O}(mp(m+p)\log p)$.*

Démonstration. L'algorithme de discrétisation s'exécute en temps polynomial et génère p fenêtres de temps. Le problème de l'existence d'un ordonnancement réalisable, formulé comme un problème de transport dans la section 3.3.1, peut être réduit à un problème de flot maximum en construisant un graphe dans lequel les sommets représentent les capteurs et les fenêtres de temps, et en comparant la valeur du flot à l'horizon de temps H. Le problème de flot maximum est résolu en $\mathcal{O}(|V||E|)$ (Orlin 2013), où $|V| = \mathcal{O}(m+p)$ est le nombre de sommets et $|E| = \mathcal{O}(mp)$ est le nombre d'arcs. Puisque la dichotomie effectue un nombre logarithmique d'itérations, la complexité finale est en $\mathcal{O}(mp(m+p)\log p)$. □

Il est possible de construire une instance pour laquelle le nombre de ticks est aussi grand que l'on veut, en faisant varier un paramètre. La figure 3.9 illustre une trajectoire de cible modélisée par une sinusoïde qui se déplace le long du bord d'une zone de couverture. Puisque la période de la sinusoïde peut être aussi petite que l'on veut, le nombre d'intersections avec le bord, donc le nombre de fenêtres de temps, peut être aussi grand que l'on veut. Par conséquent, le nombre d'itérations de l'algorithme général dépend fortement de la période de la sinusoïde.

Lemme 5. *Si la trajectoire est une ligne polygonale composée de q segments, alors le problème peut être résolu en un nombre polynomial d'itérations, en $\mathcal{O}(q^2m^3\log(qm))$.*

Démonstration. La preuve est basée sur les intersections entre les segments de la ligne polygonale et les disques (zones de couverture). Chaque segment peut couper au maximum

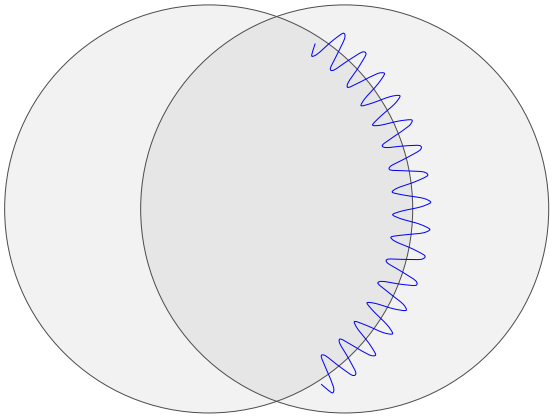


FIGURE 3.9 – Exemple de situation où le nombre de ticks est aussi grand que l'on veut

deux fois chaque cercle bordant les disques. Par conséquent, le nombre de ticks ne peut dépasser $2qm$, et le nombre de fenêtres de temps est au maximum $p = 2qm-1$. Le problème de décision, exprimé comme un problème de flot maximum, est résolu en $\mathcal{O}(|V||E|)$ (ORLIN 2013), où $|V| = \mathcal{O}(qm)$ est le nombre de sommets et $|E| = \mathcal{O}(qm^2)$ est le nombre d'arcs. Puisque la dichotomie s'exécute en un nombre d'itérations logarithmique, la complexité finale est en $\mathcal{O}(q^2m^3 \log(qm))$. □

3.3.3 Exemple

Exécutons l'algorithme 4 sur un exemple simple, similaire à celui montré sur la figure 2.3. Il comporte 3 capteurs, ayant pour coordonnées $(-3, 0)$, $(0, -5)$, $(3, 0)$. Leurs zones de couverture respectives $\{s_1, s_2, s_3\}$ sont des disques de rayon $R^S = 6$ et les capacités de leur batterie sont $E_i = 15$. La trajectoire de la cible est une ligne polygonale passant par 5 points de contrôle définis dans le tableau 3.3. La vitesse de la cible est uniforme le long de chaque segment de la ligne polygonale, mais peut changer au passage d'un point de contrôle.

Point de contrôle	1	2	3	4	5
Date	0	2	5	8	11
Coordonnées	$(-7, -2)$	$(-4, -3)$	$(0, -2)$	$(-1, 2)$	$(-3, 4)$

TABLEAU 3.3 – Dates et coordonnées des points de contrôle définissant la trajectoire de la cible

La trajectoire étant représentée comme une ligne polygonale, la discrétisation consiste à calculer des intersections entre des segments et des cercles. Cela revient ainsi à résoudre des équations linéaires du second degré à coefficients constants.

Tick	t_1	t_2	t_3	t_4	t_5	t_6
Date	0	1.026	3.195	7.216	9.685	11
Statut*	$-$	$+$	$+$	$-$	$-$	$+$
Face visitée	f_1	f_2	f_3	f_4	f_1	-

*$+$ = tick entrant, $-$ = tick sortant

TABLEAU 3.4 – Résultat de la discrétisation

La cible traverse les faces $f_1 = \{s_1\}$, $f_2 = \{s_1, s_2\}$, $f_3 = \{s_1, s_2, s_3\}$ et $f_4 = \{s_1, s_3\}$. L'ensemble des ticks et la séquence des faces sont donnés dans le tableau 3.4. Les deux bornes supérieures UB1′ et UB2′ sont calculées comme indiqué à la section 3.2.3.

$$\text{UB1}' = \min\{7.5, 8.584, 10.595, 11.829, 15, 9.51, 10.745, 15, 8.734, 7.5\} = 7.5$$

$$\text{UB2}' = \min\{3.165, 9.5, 17, 9.5\} = 3.165$$

Il en résulte que UB = 3.165 est une borne supérieure sur le rayon de stabilité. La première étape consiste à vérifier qu'il existe un ordonnancement réalisable tel que $\rho = \text{UB}$. Le problème de transport associé à l'instance $\mathcal{I}_{3.165}$ donne une valeur de fonction objectif strictement positive, donc la valeur de ρ optimale sera strictement inférieure à 3.165.

La seconde étape consiste à vérifier l'existence d'un ordonnancement réalisable pour $\rho = 0$. Puisque $\mathcal{I}_0$ est réalisable, on peut alors appliquer la dichotomie. L'ensemble ordonné des distances $t'_k - t_k < UB$ est $\mathcal{D} = \{0, 2.01, 3.095\}$. Le tableau 3.5 donne les valeurs de Δ_k associées aux différentes instances en fonction de ρ.

ρ	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5
0	1.026	2.169	4.021	2.469	1.315
2.010	5.047	2.169	0	2.469	5.336
3.095	7.216	0	2.169	0.300	7.505
3.165	7.216	0.140	2.169	0.160	7.644

TABLEAU 3.5 – Les valeurs de Δ_k en fonction de ρ

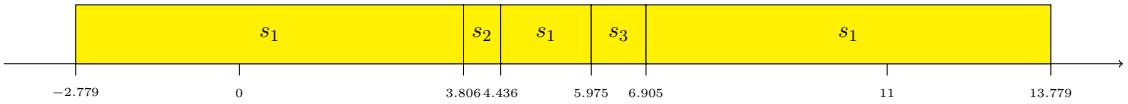


FIGURE 3.10 – Diagramme de Gantt d'un ordonnancement optimal

La dichotomie teste l'existence d'un ordonnancement réalisable pour les valeurs $\rho = 2.01$ puis $\rho = 3.095$. L'instance $\mathcal{I}_{2.01}$ est réalisable, tandis que $\mathcal{I}_{3.095}$ ne l'est pas. Par conséquent, la valeur optimale de ρ se situe dans l'intervalle $[2.01, 3.095[$. Finalement, la résolution du programme linéaire défini par les équations (3.5)-(3.9) donne $\delta = 0.769$. Donc le rayon de stabilité optimal est $\rho = 2.779$. L'ordonnancement obtenu est illustré sous la forme d'un diagramme de Gantt sur la figure 3.10. On remarque que le rayon de stabilité est limité spécifiquement par la capacité du capteur 1, dont la batterie est épuisée. La durée de la mission de surveillance associée à l'ordonnancement maximisant le rayon de stabilité est de $H + 2\rho = 16.558$ unités de temps.

3.4 Résultats

La méthode proposée dans ce chapitre a été testée sur un jeu de 200 instances générées aléatoirement. Chacune d'entre elles décrit la trajectoire d'une cible dans une région carrée de dimensions $\sqrt{10m} \times \sqrt{10m}$. Ainsi, la densité moyenne des réseaux, égale au nombre de capteurs par unité de surface, est la même pour toutes les instances. La trajectoire est une ligne polygonale composée de 9 segments reliés par 10 points de contrôle. Chaque point de contrôle est associé à un temps de passage de la cible. Ces temps sont uniformément distribués le long de l'horizon de temps $H = 10m$ où m est le nombre de capteurs. Comme les points de contrôle sont générés aléatoirement, la cible est susceptible de traverser plusieurs fois une même face. Les m capteurs ($m \in \{100, 200, 500, 1000\}$) sont positionnés aléatoirement, en s'assurant que leur zone de couverture chevauche une partie de la trajectoire. Les zones de couverture sont des disques de rayon $R^S = 10$. Plusieurs valeurs de capacité initiale $E_i \in \{12, 16, 20\}$ ont été testées pour chaque instance.

Le programme a été implémenté en C++ et exécuté sur une machine équipée d'un processeur Intel Xeon W3520 (2.67 GHz × 8) avec 8 Go de RAM sous Linux Ubuntu 14.04. Le programme linéaire défini par les équations (3.5)-(3.9) est résolu à l'aide de CPLEX 12.6.1. Le problème de décision est reformulé comme un problème de flot maximum et est résolu à l'aide de la bibliothèque LEMON 1.3.1 de COIN-OR (DEZSŐ *et al.* 2011).

Le tableau 3.6 montre les résultats numériques pour 4 catégories de 50 instances groupées par nombre de capteurs. La seconde colonne est le nombre moyen de fenêtres de temps par instance. Chaque ligne contient un quadruplet #UB1/#UB2/#∅/#Inf où #UB1 (respectivement #UB2) est le nombre d'instances pour lequel UB1′ (respectivement UB2′) est atteinte, #∅ est le nombre d'instances pour lequel aucune borne n'a été atteinte, et #Inf est le nombre d'instances non-réalisables.

m	$\|K\|$	$E_i = 12$	$E_i = 16$	$E_i = 20$
100	373.26	9/2/30/9	44/0/4/2	46/1/2/1
200	628.68	3/1/26/20	32/2/11/5	45/1/2/2
500	1465.12	0/0/6/44	12/0/26/12	38/0/10/2
1000	2707.54	0/0/8/42	11/0/23/16	29/0/18/3

TABLEAU 3.6 – Répartition entre les instances (#UB1/#UB2/#∅/#Inf) en fonction du nombre de capteurs et de leurs capacités initiales

La borne UB1′ domine UB2′ dans 93.4% des instances réalisables. UB2′ semble plus efficace lorsque la capacité initiale des capteurs est critique, c'est-à-dire lorsque la capacité est juste suffisante pour assurer la réalisabilité de l'instance. On peut en déduire que la capacité initiale des capteurs est un facteur significatif dans le temps de calcul.

Lorsqu'une des bornes supérieures est atteinte, le calcul des bornes supérieures représente en moyenne 45.8% du temps total d'exécution. La génération des instances $\mathcal{I}_\rho$ de l'algorithme 3 représente 37.6% du temps de calcul en moyenne. Pour toutes les autres instances réalisables, le calcul des bornes supérieures représente 7.58% du temps total. La génération des instances $\mathcal{I}_\rho$ représente 62.3% du temps total, tandis que résoudre le problème de transport représente 22.6% du temps.

La difficulté à résoudre le problème augmente avec la taille de l'instance, mesurable en fonction du nombre de capteurs, mais également en fonction du nombre de fenêtres de temps. Néanmoins, on peut observer dans le tableau 3.6 que les capacités initiales des capteurs jouent un rôle majeur vis-à-vis des bornes supérieures. Or, ces bornes ont un impact important sur le temps de calcul, comme le montre le tableau 3.7. Pour des faibles capacités initiales, la plupart des instances sont non-réalisables. Cette situation se présente notamment pour les instances pour lesquelles la cible se déplace sur une plus grande surface. Au contraire, lorsque la capacité initiale des capteurs est élevée, la plupart des instances est réalisable. Plus spécifiquement, UB1′ est atteint dans la plupart des cas, en particulier sur les petites instances. Ceci est dû au fait que lorsque les capacités initiales sont élevées,

le rayon de stabilité est rarement limité par celles-ci, mais par la durée minimale entre deux fenêtres de temps ayant peu de capteurs candidats en commun. Pour les cas intermédiaires ($E_i = 16$), aucune borne n'est atteinte dans la majorité des cas, bien que la plupart des instances soient réalisables. Dans ces cas, la dichotomie est utilisée. Par conséquent, ces résultats montrent que le problème est généralement facile pour les cas extrêmes (capacités initiales suffisamment faibles ou suffisamment élevées), et que l'écart entre ces cas extrêmes est relativement serré.

m	$\min\{\text{UB1}', \text{UB2}'\}$	$\emptyset$
100	0.086 s	0.481 s
200	0.215 s	1.206 s
500	1.311 s	8.652 s
1000	5.252 s	33.55 s

TABLEAU 3.7 – Moyenne du temps d'exécution sur des instances réalisables dans le cas où les bornes UB1′ or UB2′ sont atteintes comparé au cas où elles ne le sont pas

Le tableau 3.7 montre le temps de calcul moyen lorsque UB1′ ou UB2′ sont atteintes (seconde colonne) et lorsque aucune d'entre elles ne l'est (troisième colonne). Seules les instances réalisables sont prises en compte. Le tableau 3.7 montre l'impact significatif des bornes sur la résolution. Les instances réalisables pour lesquelles UB1′ ou UB2′ sont optimales sont en moyenne 5 à 7 fois plus rapides à résoudre que le reste des instances réalisables. Ceci est dû au fait que l'algorithme 4 s'arrête avant d'avoir démarré la dichotomie.

Le tableau 3.8 donne les temps de calcul et les valeurs optimales moyennes de ρ en fonction du nombre de capteurs et de leurs capacités initiales. La colonne CPU_0 contient le temps moyen pour atteindre le début de la dichotomie, c'est-à-dire le moment où est trouvé le premier ordonancement réalisable pour $\rho = 0$. La colonne CPU_ρ affiche le temps moyen pour obtenir la solution optimale.

Un ordonnancement avec un rayon de stabilité nul, comme celui trouvé avant le début de la dichotomie, consomme moins d'énergie qu'un ordonnancement optimal. Ceci est dû au fait que les capteurs doivent être activés ρ unités de temps avant la date d'apparition prévisionnelle de la cible et ρ unités de temps après la date d'arrivée prévue. Donc le coût énergétique additionnel d'un ordonnancement de rayon de stabilité ρ est égal à 2ρ. Dans nos instances, ce coût représente en moyenne 4.1% de l'horizon de temps, et 16.1% dans le pire des cas.

m	$\|K\|$	E_i	CPU_0	CPU_ρ	Avg. ρ
100	373.26	12	0.076 s	0.321 s	25.9
		16	0.074 s	0.112 s	34.8
		20	0.074 s	0.106 s	35.5
200	628.68	12	0.183 s	0.717 s	25.1
		16	0.184 s	0.432 s	45.9
		20	0.184 s	0.248 s	49.2
500	1465.12	12	1.106 s	1.804 s	26.3
		16	1.159 s	5.342 s	64.3
		20	1.138 s	2.754 s	78.4
1000	2707.54	12	4.311 s	10.47 s	48.5
		16	4.359 s	17.13 s	86.3
		20	4.276 s	14.70 s	106.2

TABLEAU 3.8 – Moyenne du temps d'exécution en fonction du nombre de capteurs et de leurs capacités initiales

Le temps de calcul requis pour obtenir une solution optimale dépend significativement de l'instance. Lorsque l'instance est non-réalisable, ou lorsqu'une borne supérieure est atteinte par la valeur de ρ optimale, alors CPU_ρ est égal à CPU_0, puisque l'algorithme s'arrête avant la dichotomie. Sinon, CPU_ρ peut être significativement supérieur à cause de la dichotomie.

3.5 Conclusion

Dans ce chapitre, un problème de robustesse a été étudié. L'approche proposée calcule un ordonnancement des activités de capteurs dont le rayon de stabilité est maximum. La méthode se déroule en deux phases. La première phase transforme les données d'entrée afin de les présenter sous une forme adéquate permettant de formuler le problème comme un problème d'ordonnancement d'activités de surveillance. Puis, la seconde phase repose sur une dichotomie où chaque itération consiste à tester l'existence d'un ordonnancement réalisable pour un rayon de stabilité donné. L'algorithme général s'exécute en un nombre pseudo-polynomial d'itérations en fonction du nombre de capteurs et du nombre de ticks générés. La complexité devient polynomiale lorsque la trajectoire de la cible est représentée par une ligne polygonale. Les résultats numériques ont montré que l'algorithme résout des instances comportant jusqu'à 1000 capteurs en moins de 15 secondes en moyenne. Les

bornes supérieures accélèrent significativement l'algorithme, puisque celles-ci sont souvent atteintes et permettent d'éviter le lancement de la dichotomie. Les résultats ont également montré l'impact des capacités initiales des batteries sur le temps de calcul requis par notre approche. Les travaux réalisés dans ce chapitre ont fait l'objet d'une publication (Lersteau *et al.* 2016).

Le cas particulier étudié dans ce chapitre constitue d'ores et déjà un point de départ à de nombreuses perspectives. Une extension naturelle est de considérer le cas impliquant plusieurs cibles à surveiller. En restant dans le cas d'une seule cible à surveiller, on peut considérer des variantes où plusieurs capteurs peuvent être activés simultanément, de telle sorte à augmenter davantage le rayon de stabilité. Enfin, il pourrait être intéressant d'étudier le compromis entre la maximisation du rayon de stabilité, et la minimisation de l'énergie consommée.

Chapitre 4

Optimisation de la consommation énergétique

Contenu

4.1 Introduction

L'un des avantages majeurs des réseaux de capteurs sans fil est de pouvoir surveiller des cibles dans une vaste zone, à moindre coût. En effet, les réseaux de capteurs sans fil sont typiquement constitués d'appareils bon marché, qu'on peut massivement déployer dans une zone où les infrastructures de surveillance sont inadaptées ou inexistantes, en les larguant d'un avion ou d'un hélicoptère. Les capteurs, autonomes, sont généralement alimentés par une batterie dont la durée de vie est limitée. Par conséquent, la consommation énergétique est un aspect important des missions de surveillance. En général, il n'est pas utile d'activer tous les capteurs en même temps pour surveiller toutes les cibles, dans la mesure où de nombreuses parties de la zone à surveiller sont couvertes par plusieurs capteurs en même temps. En supposant que les cibles, ainsi que leurs trajectoires, sont connues à l'avance, notre but est de planifier un ordonnancement d'activités des capteurs. Cet ordonnancement doit être tel qu'il préserve autant que possible la capacité courante du réseau de capteurs à surveiller ultérieurement certaines zones jugées intéressantes par l'exploitant du réseau de capteurs. On souhaite également que l'ordonnancement minimise l'énergie totale consommée pendant la mission de surveillance. Le contexte d'utilisation du réseau de capteurs est désormais celui de missions multiples, où l'on cherche d'abord à préserver le réseau pour de futures missions de surveillance, tout en privilégiant les solutions qui minimisent l'énergie consommée pour accomplir la mission courante.

Étant donnée une zone d'intérêt incluse dans la zone de surveillance, on cherchera dans un premier temps à maximiser la *garantie de couverture*. La garantie de couverture est la durée minimale, à l'issue de la mission courante, durant laquelle le réseau de capteurs est capable d'assurer la surveillance de tout point de la zone d'intérêt. Puis, ayant fixé la garantie de couverture à sa valeur maximale, l'objectif est désormais de minimiser l'énergie totale consommée par les capteurs pour accomplir la mission de surveillance courante. Nous considérons d'abord le cas où la trajectoire des cibles est connue à l'avance, puis nous supposons qu'elles sont sujettes à incertitude, c'est-à-dire que leurs positions réelles peuvent s'écarter d'une certaine distance par rapport à leurs positions prévisionnelles.

Le reste du chapitre est organisé de la manière suivante. La section 4.2 introduit le problème général, ainsi que sa décomposition. Elle comprend les sous-sections 4.2.1 et 4.2.2 qui présentent respectivement la phase de discrétisation, et une étude de complexité, montrant deux cas particuliers où le problème de ce chapitre se résout en temps polynomial. La section 4.3 donne les formulations mathématiques des trois sous-problèmes, et introduit des

bornes supérieures. Dans la section 4.4, la méthode de résolution est détaillée. Les résultats numériques sont présentés dans la section 4.5 et la section 4.6 conclut ce chapitre.

4.2 Description du problème

Un ensemble $I = \{1, \ldots, m\}$ de m capteurs statiques est disséminé aléatoirement dans une région. Chaque capteur $i \in I$ est alimenté par une batterie d'une capacité initiale E_i exprimée en unités de temps, et peut surveiller des cibles dans sa zone de couverture s_i. Sans perte de généralité, nous supposons que la zone de couverture de chaque capteur est définie par un disque de rayon R^S. Lorsqu'un capteur i est actif, il surveille toutes les cibles se situant dans la zone de couverture s_i et puise de l'énergie sur sa batterie indépendamment du nombre de cibles effectivement surveillées. Lorsque le capteur est inactif, la capacité de sa batterie reste inchangée. L'union des zones de couverture est appelée zone de surveillance.

I	Ensemble des capteurs $\{1, \ldots, m\}$
J	Ensemble des cibles $\{1, \ldots, n\}$
s_i	Zone de couverture du capteur i
E_i	Capacité du capteur i
$\mathcal{T}_j(t)$	Position de la cible j à l'instant t
$Z \subseteq \cup_{i \in I} s_i$	Zone d'intérêt

Tableau 4.1 – Données d'entrée du problème

Un ensemble de n cibles $J = \{1, \ldots, n\}$ se déplace dans la zone de surveillance. Sans perte de généralité, nous faisons l'hypothèse que les cibles évoluent sur une surface plane, et que les données géométriques du problème sont en deux dimensions. Leur trajectoire est décrite par une fonction vectorielle continue $\mathcal{T}_j : \mathbb{R} \to \mathbb{R}^2$, définie par $\mathcal{T}_j(t)$ où t est une variable de temps. Sans perte de généralité, on considère que t a pour domaine $[0, H]$ où H est la durée prévisionnelle de la mission, ou horizon de temps. Les trajectoires spatiales des cibles sont sujettes à incertitude. À chaque instant t, la position d'une cible j peut s'écarter d'une distance δ de sa position attendue $\mathcal{T}_j(t)$. On considère alors pour chaque cible un disque d'incertitude à couvrir, de rayon δ et centré sur le point $\mathcal{T}_j(t)$. La valeur de δ peut varier au cours du temps. En effet, on peut considérer par exemple que δ est nul à $t = 0$ quand la mission débute, et qu'il est proportionnel à t, afin de traduire le fait que l'incertitude spatiale augmente avec le temps. On considère $Z \subseteq \cup_{i \in I} s_i$, une zone d'intérêt, sous-ensemble de la zone de surveillance qu'on souhaite pouvoir surveiller davantage après

la mission. Le tableau 4.1 contient les données d'entrée du problème traité dans ce chapitre.

La mission doit répondre aux contraintes suivantes :

— la somme des activités de surveillance d'un capteur ne peut dépasser la capacité de sa batterie.

— chaque cible doit être surveillée par au moins un capteur à chaque instant de la mission.

Les activités de surveillance sont planifiées avant la mission, et leur ordonnancement doit répondre à certains critères. En particulier, l'ordonnancement doit assurer une *garantie de couverture* maximale. Une garantie de couverture d'une valeur $T_{\min}$ assure que le réseau de capteurs est capable de surveiller n'importe quel point de la zone d'intérêt Z durant $T_{\min}$ unités de temps supplémentaires à l'issue de la mission courante. En effet, chaque point de Z possède un potentiel de surveillance, qui correspond à la somme des capacités restantes des batteries des capteurs le couvrant. Par conséquent, maximiser la garantie de couverture revient à maximiser le plus petit potentiel de surveillance. Enfin, il peut exister une multitude d'ordonnancements conduisant à une garantie de couverture maximale. L'ordonnancement retenu doit assurer que la consommation d'énergie induite par la mission de surveillance courante est minimale, tout en assurant une garantie de couverture maximale.

Le problème général se décompose alors en trois sous-problèmes successifs. Le premier, noté MRC (pour Maximize Residual Capacity), consiste à vérifier l'existence d'un ordonnancement réalisable. La nécessite de traiter ce sous-problème explicitement sera justifiée aux sections 4.2.2 et 4.4.3. Le second sous-problème, noté MCG (pour Maximize Coverage Guarantee), consiste à déterminer la garantie de couverture maximale du réseau de capteurs. Le troisième sous-problème, noté MEC (pour Minimize Energy Consumption), consiste à minimiser l'énergie totale consommée par les capteurs pour remplir la mission courante, en assurant la garantie de couverture maximale déterminée par MCG.

La résolution du problème général se déroule en deux phases. La première phase est la discrétisation, elle vise à mettre en forme les données du problème, tandis que la seconde phase consiste à résoudre successivement les trois sous-problèmes mentionnés ci-dessus, afin d'ordonnancer l'activité des capteurs en s'appuyant sur les résultats de la discrétisation.

4.2.1 Discrétisation

À notre connaissance, aucune méthode ne permet de résoudre directement ce problème à partir des données géométriques. C'est pourquoi nous faisons appel à une étape préliminaire

de reformulation. Celle-ci est expliquée en détail dans le chapitre 2 et résumée dans cette section pour notre problème. La zone de surveillance peut être partionnée en un ensemble de faces $\widehat{F}$, où une face est un ensemble de points couvert par les mêmes capteurs (Berman *et al.* 2004 ; Meguerdichian *et al.* 2003 ; Slijepcevic *et al.* 2001).

Dans un premier temps, nous supposons que la position des cibles n'est pas sujette à incertitude. Notons $F \subseteq \widehat{F}$ l'ensemble des faces traversées par les cibles. La trajectoire empruntée par une cible peut être décrite comme une séquence de faces, où chaque élément de la séquence est associé à une durée de séjour. Pour surveiller une cible, il suffit d'activer l'un des capteurs couvrant la face dans laquelle elle est située. Ainsi, la position exacte des cibles n'est plus nécessaire pour déterminer un ordonnancement d'activités de capteurs. On appelle *ticks* les instants où une cible franchit la frontière d'une face, et *fenêtres de temps* les intervalles de temps entre deux ticks consécutifs. On note K l'ensemble des fenêtres de temps, de cardinalité p. Leurs durées respectives sont notées Δ_k et les ticks sont notés t_k. Dans ce chapitre, la variante de la discrétisation utilisée est celle suivant les faces, décrite dans la section 2.4 (page 24). L'ensemble des capteurs candidats pour la couverture d'une face f est noté $S(f)$. L'ensemble des faces à couvrir durant la fenêtre de temps k est noté $T(k)$. Les faces chevauchant la zone d'intérêt Z forment un ensemble noté $F^* \subseteq \widehat{F}$. Le tableau 4.2 résume le résultat de la discrétisation.

F	Ensemble des faces $\{f_1, \ldots, f_q\}$ visitées par au moins une cible
F^*	Ensemble des faces d'intérêt $\{f_1, \ldots, f_{q'}\}$
K	Ensemble des fenêtres de temps $\{1, \ldots, p\}$
$S(f) \subseteq I$	Ensemble des capteurs candidats couvrant la face $f \in F \cup F^*$
$T(k) \subseteq F$	Ensemble des faces à couvrir pendant la fenêtre de temps $k \in K$
t_k	Tick d'indice $k \in K \cup \{p+1\}$
Δ_k	Durée de la fenêtre de temps $k \in K$

Tableau 4.2 – Données résultant de la discrétisation suivant les faces

Considérons le cas où la position des cibles est sujette à incertitude, c'est-à-dire qu'elles peuvent s'écarter d'une distance δ de leur position attendue. Afin d'être certain de couvrir chaque cible malgré l'incertitude, les capteurs doivent couvrir un disque de rayon δ centré sur la position attendue de la cible (figure 4.1). Contrairement aux cibles ponctuelles, ce disque peut chevaucher plusieurs faces simultanément. Il s'ensuit que le nombre de faces à couvrir, et donc le nombre de capteurs à activer, peut potentiellement augmenter. Toutefois, cela ne modifie pas la structure des données résultant de la discrétisation. Les

ticks correspondent aux temps pour lesquels le disque d'incertitude rencontre la frontière d'une face ou cesse d'en chevaucher une autre.

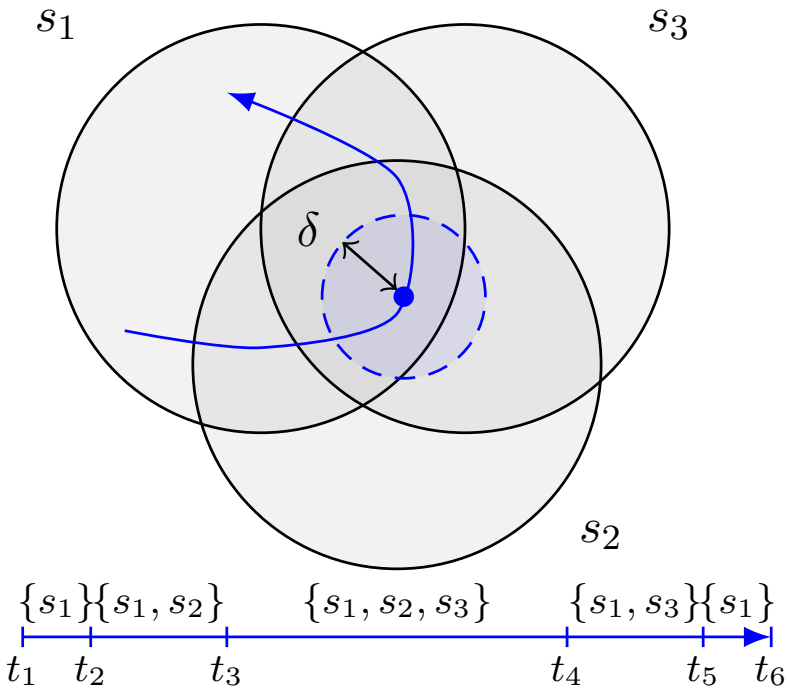


FIGURE 4.1 – Positions potentielles d'une cible sous incertitude

Soit une fenêtre de temps d'indice k. Une *couverture* valide pour la fenêtre de temps k est un sous-ensemble de capteurs capable de couvrir toutes les faces de $T(k)$. Une couverture est donc valide si elle contient, pour chaque face $f \in T(k)$, au moins un capteur de $S(f)$. Ainsi, à partir des données de la discrétisation, les sous-problèmes MRC, MCG et MEC peuvent être vus comme des problèmes d'ordonnancement d'activités de couvertures.

De plus, ces données permettent de définir autrement la notion de garantie de couverture. Chaque face de la zone de surveillance a un potentiel de surveillance, c'est-à-dire un temps durant lequel les capteurs candidats sont capables de la surveiller. La garantie de couverture est ainsi définie par la valeur minimale du potentiel de surveillance de toutes les faces d'intérêt. Par conséquent, une garantie de couverture d'une valeur $T_{\min}$ assure que l'on peut surveiller n'importe quelle face de F^* pendant une durée au moins égale à $T_{\min}$ unités de temps. En revanche, elle ne garantit pas la possibilité de surveiller plusieurs faces simultanément pendant $T_{\min}$ unités de temps.

4.2.2 Étude de complexité

Les notions introduites par la discrétisation permettent d'étudier la complexité du problème général présenté dans ce chapitre, ainsi que des cas particuliers où le problème admet un algorithme polynomial.

Lemme 6. *Déterminer si MRC est réalisable est un problème NP-complet.*

Démonstration. La preuve se déroule en trois étapes. La première consiste à montrer que la taille d'une solution est bornée polynomialement par la taille du problème. La seconde consiste à montrer qu'on peut vérifier l'admissibilité d'une solution en temps polynomial. Enfin, la dernière étape consiste à montrer qu'un cas particulier de MRC est NP-complet.

1. MRC est modélisé comme un programme linéaire constitué de $2m + |K|$ contraintes hors contraintes de non-négativité (Section 4.3.1). Le nombre de variables de base étant égal au nombre de contraintes, il ne peut y avoir plus de $2m + |K|$ couvertures dont la durée d'activation est strictement positive. À chaque couverture est associé un tableau de m booléens et une durée, soit $m + 1$ nombres. Par conséquent, une solution de MRC est constituée d'au plus $(2m + |K|)(m + 1)$ nombres, ce qui est une quantité polynomiale en la taille du problème. Pour MCG et MEC, le raisonnement est identique, hormis que les programmes linéaires contiennent au plus $m+|K|+|F| \leq m+|K|+m(m-1)+2$ contraintes. Une solution de MCG et MEC est alors constituée d'au plus $(m^2+|K|+2)(m+1)$ nombres. Donc la taille d'une solution est bornée polynomialement par la taille du problème.

2. Pour vérifier qu'une solution de MRC, MCG ou MEC est réalisable, on initialise u_i la durée d'activation totale de chaque capteur i à zéro et v_k le temps de surveillance dans la fenêtre de temps k à zéro également. On parcourt une seule fois chaque couverture $c \in \mathcal{C}$ de la solution (leur nombre est polynomial). Pour chacune d'entre elles, on note k la fenêtre de temps et d_c^k la durée d'activation correspondantes. On incrémente de d_c^k la valeur de v_k. Si la couverture c contient le capteur i, on incrémente de d_c^k la valeur de u_i. Une fois la solution totalement parcourue, on vérifie que $u_i \leq E_i$ (contrainte 4.14) et $v_k = \Delta_k$ (contrainte 4.15). Si oui, alors la solution est réalisable. Le nombre total d'opérations effectuées est égal au nombre de couvertures activées (voir point 1.) plus m plus $|K|$, ce qui est polynomial en la taille de l'instance.

3. Les points 1. et 2. prouvent que MRC est dans NP. Pour montrer que MRC est NP-difficile, on construit une instance de MRC constituée d'une seule fenêtre de temps de durée Δ. Dans ce cas, déterminer si MRC est réalisable équivaut à résoudre le problème de décision de MNLB (Rossi *et al.* 2011), prouvé NP-complet, qui consiste à déterminer

s'il existe un ordonnancement tel que la durée de vie du réseau est supérieure ou égale à Δ. □

Ce lemme s'applique également à MCG et MEC. On en déduit que MRC, MCG et MEC sont des problèmes NP-difficiles, et le lemme 6 justifie la nécessité de savoir si le problème est réalisable avant de chercher à déterminer la garantie de couverture maximale (MCG) puis d'obtenir une solution minimisant l'énergie consommée pour remplir la mission de surveillance courante en assurant cette garantie de couverture maximale (MEC). Néanmoins, la procédure de réduction d'instance décrite dans la section 2.4.2 est susceptible de réduire les temps de calcul en pratique. Cette procédure consiste à retirer des faces redondantes des ensembles $T(k)$, puis à fusionner les fenêtres de temps qui partagent le même ensemble de faces à couvrir.

Soit F l'ensemble de toutes les faces qui sont visitées par au moins une cible pour une durée strictement positive. Pour tout $\Omega \subseteq F$, on définit $H(\Omega) = \bigcup_{f \in \Omega} S(f)$ l'ensemble des capteurs qui peuvent couvrir au moins une face dans Ω.

Lemme 7. (Condition suffisante de non-réalisabilité)
S'il existe $\Omega \subseteq F$ *tel que* $\sum_{i \in H(\Omega)} E_i < \sum_{k \in K | \Omega \cap T(k) \neq \emptyset} \Delta_k$, *alors MRC est non-réalisable.*

Démonstration. Si le temps nécessaire de surveillance d'un ensemble de faces dépasse la durée maximale d'activation des capteurs candidats, le problème est alors non-réalisable. □

Lemme 8. (Condition suffisante de réalisabilité)
Soit $K(i) = \{k \in K | \exists f \in T(k), i \in S(f)\}$ *l'ensemble des fenêtres de temps durant lesquelles le capteur* i *est candidat. Si* $\exists i \in S(f), E_i \geq \sum_{k \in K(i)} \Delta_k$ *pour toute face* $f \in F$, *alors MRC est réalisable.*

Démonstration. Si, pour chaque face, il existe un capteur qui peut la surveiller (ainsi que toute autre face incluse dans sa zone de couverture) pour une durée supérieure ou égale à la durée de surveillance requise, alors une solution réalisable peut être construite en activant ce capteur chaque fois que la face a besoin d'être couverte. □

Remarque. Les lemmes 7 and 8 s'appliquent également à MCG et MEC.

Dans les deux cas particuliers suivants, MRC, MCG et MEC peuvent être résolus de manière optimale (ou prouvés non-réalisables) en temps polynomial. Dans le premier cas,

toutes les cibles peuvent être couvertes par n'importe quel capteur à chaque instant. Ce cas correspond à des situations où les zones de couverture des capteurs sont très larges. À l'inverse, le second cas correspond à des situations où les zones de couverture sont si petites qu'aucun capteur ne peut couvrir plus d'une seule cible à la fois. Dans les deux cas, MCG est formulé comme un programme linéaire de taille raisonnable. De plus, la solution optimale de MCG est prouvée optimale pour MEC. Par conséquent, la résolution de MCG et MEC dans ces cas particuliers peut s'effectuer en temps polynomial.

Cas des capteurs à longue portée

Dans un premier temps, considérons le cas où chaque capteur est capable de couvrir toutes les cibles à chaque instant. Cette situation se présente lorsque les capteurs ont une portée très longue. Dans ce cas, la valeur optimale de l'objectif de MEC, qui est l'énergie dépensée pour accomplir la mission de surveillance courante, est H puisqu'un seul capteur est utilisé à chaque instant. De plus, la discrétisation n'est pas nécessaire puisque toutes les cibles se déplacent ensemble, à l'intérieur de la même face. Dans ces circonstances, MCG est formulé par le programme linéaire noté $\mathrm{LP}_{\mathrm{long}}$. Soit x_i une variable continue non-négative représentant le temps total d'activation du capteur i. La variable $T_{\min}$ est la garantie de couverture offerte pour les faces de F^*.

Modèle $\mathrm{LP}_{\mathrm{long}}$

$$\max\ T_{\min} \tag{4.1}$$

$$\sum_{i \in I} x_i = H \tag{4.2}$$

$$x_i \leq E_i \qquad \forall i \in I \tag{4.3}$$

$$\sum_{i \in S(f)} (E_i - x_i) \geq T_{\min} \qquad \forall f \in F^* \tag{4.4}$$

$$T_{\min} \geq 0 \tag{4.5}$$

$$x_i \geq 0 \qquad \forall i \in I \tag{4.6}$$

La contrainte (4.2) impose que la durée de la mission soit de H unités de temps, et la contrainte (4.3) assure que la durée d'activation d'un capteur ne dépasse pas sa capacité. La contrainte (4.4) borne la garantie de couverture des faces d'intérêt. Le programme linéaire $\mathrm{LP}_{\mathrm{long}}$ est constitué de $\mathcal{O}(m)$ variables et $\mathcal{O}(m^2)$ contraintes, puisque le nombre de faces

est inférieur à $m(m-1)+2$ (Berman *et al.* 2004). Comme la taille de ce programme linéaire n'augmente pas de manière exponentielle avec la taille des données, il peut être résolu par un solveur sans technique de décomposition ou traitement préalable.

$\mathrm{LP}_{\mathrm{long}}$ est non-réalisable si et seulement si $\sum_{i\in I} E_i < H$. Dans ce cas, les capteurs n'ont pas suffisamment d'énergie pour couvrir les cibles pendant H unités de temps, et donc MRC, MCG et MEC sont non-réalisables. Si $\mathrm{LP}_{\mathrm{long}}$ est réalisable, alors la solution optimale permet de construire un ordonnancement optimal pour MRC, MCG et MEC. Les capteurs peuvent être activés dans l'ordre que l'on souhaite, et chaque capteur i est activé x_i unités de temps, $\forall i \in I$. Cette solution est optimale pour MEC puisque la contrainte (4.2) garantit que l'énergie totale dépensée durant la mission est minimale.

Cas des capteurs à courte portée

Considérons maintenant le cas où les cibles sont positionnées de telle sorte qu'aucun capteur ne peut couvrir plus d'une cible à la fois. Une condition suffisante pour que cette situation se présente est que la distance entre chaque paire de cibles demeure toujours strictement supérieure à $2R^S$, où R^S est le rayon des zones de couverture des capteurs. Dans ce cas particulier, la valeur optimale de l'objectif de MEC est nH. En effet, puisqu'aucun capteur ne peut couvrir plus d'une cible à la fois, alors exactement n capteurs doivent être actifs à chaque instant. Par conséquent, l'énergie nécessaire à la mission est toujours égale à nH.

Soit $F^j \subseteq F$ l'ensemble des faces visitées par la cible j, $\forall j \in J$. Notons que si une face est visitée à plusieurs reprises par une cible j, alors elle n'apparaît qu'une seule fois dans F^j. La durée totale de séjour d'une cible j dans une face f est notée Δ^{jf} pour tout $f \in F^j$. Bien que $F^j \cap F^{j'}$ ne soit pas nécessairement vide pour $j \neq j'$, il faut observer que les faces de $F^j \cap F^{j'}$ ne sont jamais visitées simultanément par les cibles j et j', étant données les hypothèses de départ.

Soit x_i^{jf} une variable continue non-négative représentant le temps pendant lequel le capteur i couvre la face f pour surveiller la cible j. La variable $T_{\min}$ est la garantie de couverture offerte pour les faces de F^*.

La contrainte (4.8) impose que chaque face est couverte pendant la durée requise, et la contrainte (4.9) assure que la durée totale d'activation d'un capteur ne dépasse pas sa capacité. La garantie de couverture est bornée grâce à la contrainte (4.10).

Si le programme linéaire $\mathrm{LP}_{\mathrm{short}}$ est non-réalisable, alors MRC, MCG et MEC sont également non-réalisables. Si $\mathrm{LP}_{\mathrm{short}}$ est réalisable, alors la solution optimale permet de

Modèle $\mathrm{LP}_{\mathrm{short}}$

$$\max\ T_{\min} \tag{4.7}$$

$$\sum_{i \in S(f)} x_i^{jf} = \Delta^{jf} \qquad \forall j \in J, \forall f \in F^j \tag{4.8}$$

$$\sum_{j \in J} \sum_{f \in F^j} x_i^{jf} \leq E_i \qquad \forall i \in I \tag{4.9}$$

$$\sum_{i \in S(f)} \left(E_i - \sum_{j \in J} \sum_{f' \in F^j | i \in S(f')} x_i^{jf'} \right) \geq T_{\min} \qquad \forall f \in F^* \tag{4.10}$$

$$T_{\min} \geq 0 \tag{4.11}$$

$$x_i^{jf} \geq 0 \qquad \forall j \in J, f \in F^j, \forall i \in I \tag{4.12}$$

construire un ordonnancement optimal pour MRC, MCG et MEC. Pour chaque cible j et chaque face $f \in F^j$, un capteur de $S(f)$ tel que $x_i^{jf} > 0$ est activé pendant x_i^{jf} unités de temps. Si $\Delta^{jf} > x_i^{jf}$, alors un autre capteur i' tel quel $x_{i'}^{jf} > 0$ est activé. Les capteurs peuvent être activés dans n'importe quel ordre, puisque le problème d'ordonnancement est préemptif (on rappelle qu'une face peut être visitée plusieurs fois par une même cible). L'optimalité peut être vérifiée pour MEC en effectuant la somme de la contrainte (4.8) pour tout j (le membre de droite doit valoir H pour tout j, puisque chaque cible doit être surveillée H unités de temps). De plus, en sommant les n équations obtenues, l'énergie totale dépensée par le réseau de capteurs est égale à nH.

4.3 Formulation mathématique

La formulation mathématique proposée pour les sous-problèmes MRC, MCG et MEC repose sur la notion de couverture, introduite dans la section 4.2.1 (page 62). Pour rappel, une *couverture* valide pour une fenêtre de temps k est un sous-ensemble de capteurs capable de couvrir toutes les faces de $T(k)$. MRC, MCG et MEC sont formulés à l'aide de programmes linéaires notés respectivement MPMRC, MPMCG et MPMEC, dont le rôle est d'affecter des durées d'activation aux couvertures. Néanmoins, le nombre de couvertures valides est généralement extrêmement grand. Par conséquent, chacun de ces programmes linéaires est la formulation d'un problème maître pour un algorithme de génération de

colonnes décrit dans la section 4.4. Ils ont été formulés de manière à partager le même ensemble de couvertures valides, ainsi que le même problème auxiliaire pour la génération des colonnes. Ceci procure l'avantage de pouvoir enchaîner la résolution du sous-problème suivant en se servant des couvertures du précédent.

En supplément, un programme non-linéaire et un programme linéaire en variables mixtes pour MCG sont décrits dans la section 4.3.4. Ils servent de référence pour une comparaison avec l'approche à base de génération de colonnes.

4.3.1 Sous-problème MRC

Le sous-problème MRC a pour finalité de construire un ensemble de couvertures réalisable, ou d'apporter la preuve qu'il n'en existe pas. La résolution de MRC est justifiée puisque la formulation de MCG présentée ultérieurement nécessite un tel ensemble de couvertures. L'objectif de MRC est de maximiser la plus petite des capacités résiduelles des capteurs. La *capacité résiduelle* d'un capteur est la capacité qu'il lui reste à l'issue de la mission courante.

Soit $\mathcal{C}$ l'ensemble de toutes les couvertures et $\mathcal{C}(k)$ les couvertures réalisables pour la fenêtre de temps $k \in K$. Soit a_{ic} une constante binaire égale à 1 si le capteur $i \in I$ appartient à la couverture $c \in \mathcal{C}$, 0 sinon. Les variables de décision du programme linéaire MPMRC modélisant MRC sont les suivantes :

- d_c^k exprime la durée allouée à la couverture $c \in \mathcal{C}(k)$ durant la fenêtre de temps $k \in K$.
- r_i est la capacité résiduelle du capteur $i \in I$
- $r_{\min}$ est la plus petite capacité résiduelle

Les variables duales associées aux contraintes sont notées entre crochets.

La contrainte (4.14) vérifie que la durée d'activation d'un capteur ne dépasse pas la capacité de sa batterie. La contrainte (4.15) assure que les cibles sont couvertes continuellement durant chaque fenêtre de temps. Enfin, la contrainte (4.16) borne la valeur de $r_{\min}$ à la plus petite des capacités résiduelles. Notons que la variable r_i prend ses valeurs dans l'ensemble $\mathbb{R}$. MRC est réalisable si et seulement si la valeur optimale de $r_{\min}$ est positive ou nulle. Ainsi, il n'est pas nécessaire d'obtenir la valeur optimale de $r_{\min}$ puisque la non-négativité de $r_{\min}$ équivaut au caractère réalisable des trois sous-problèmes.

Modèle MPMRC

$$\max r_{\min} \tag{4.13}$$

$$\sum_{k\in K}\sum_{c\in\mathcal{C}(k)} a_{ic}d_c^k + r_i = E_i \qquad \forall i \in I \quad [\pi_i] \tag{4.14}$$

$$\sum_{c\in\mathcal{C}(k)} d_c^k = \Delta_k \qquad \forall k \in K \quad [\mu_k] \tag{4.15}$$

$$r_i \geq r_{\min} \qquad \forall i \in I \tag{4.16}$$

$$d_c^k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \tag{4.17}$$

$$r_i \in \mathbb{R} \qquad \forall i \in I \tag{4.18}$$

$$r_{\min} \in \mathbb{R} \tag{4.19}$$

4.3.2 Sous-problème MCG

Pour rappel, dans la zone de surveillance, chaque face possède un potentiel de surveillance, défini par la somme des capacités de tous les capteurs candidats. La garantie de couverture est alors définie par la valeur minimale du potentiel de surveillance de toutes les faces d'intérêt. Ainsi, on assure de pouvoir surveiller n'importe quelle face de F^* pendant la garantie de couverture, d'une durée de $T_{\min}$ unités de temps. En revanche, on ne garantit pas de pouvoir surveiller plusieurs faces d'intérêt simultanément pendant cette durée. L'objectif de MCG est d'obtenir la garantie de couverture maximale $T^*_{\min}$, afin de fixer cette garantie dans le sous-problème MEC.

Les ensembles des couvertures valides $\mathcal{C}$ et $\mathcal{C}(k)$ sont identiques à ceux introduits pour le sous-problème MRC. Soit a_{ic} une constante binaire égale à 1 si le capteur $i \in I$ appartient à la couverture $c \in \mathcal{C}$, 0 sinon. Les variables de décision du programme linéaire MPMCG modélisant MCG sont les suivantes :

- d_c^k exprime la durée allouée à la couverture $c \in \mathcal{C}(k)$ durant la fenêtre de temps $k \in K$.
- r_i est la capacité résiduelle du capteur $i \in I$
- $T_{\min}$ est la garantie de couverture à maximiser

La contrainte (4.21) assure que la variable $T_{\min}$ est égale à la garantie de couverture. Autrement, MPMRC et MPMCG partagent les mêmes contraintes (4.14)-(4.15), qui assurent la couverture des cibles et le non-dépassement des capacités des capteurs. Toutes les couvertures valides pour MPMRC le sont donc pour MPMCG. À la différence de MPMRC, dans

Modèle MPMCG

$$\max T_{\min} \tag{4.20}$$

$$(4.14) - (4.15)$$

$$\sum_{i \in S(f)} r_i \geq T_{\min} \qquad \forall f \in F^* \quad [\lambda_f] \tag{4.21}$$

$$T_{\min} \geq 0 \tag{4.22}$$

$$r_i \geq 0 \qquad \forall i \in I \tag{4.23}$$

$$(4.17)$$

MPMCG, les capacités résiduelles ne peuvent pas être négatives. La résolution de MPMRC est nécessaire, puisque la résolution de MPMCG sans contrainte de non-négativité sur les variables r_i ne garantirait pas la réalisabilité de la solution optimale. Autrement dit, il serait possible d'obtenir un ordonnancement dans lequel l'activité des capteurs excèderait leur capacité.

4.3.3 Sous-problème MEC

Le sous-problème MCG peut admettre plusieurs solutions optimales, correspondant à des ordonnancements qui peuvent requérir des consommations énergétiques différentes.

Les ensembles des couvertures valides $\mathcal{C}$ et $\mathcal{C}(k)$ sont identiques à ceux introduits pour le sous-problème MCG. Soit a_{ic} une constante binaire égale à 1 si le capteur $i \in I$ appartient à la couverture $c \in \mathcal{C}$, 0 sinon. Les variables de décision du programme linéaire MPMEC modélisant MEC sont les suivantes :

— d_c^k exprime la durée allouée à la couverture $c \in \mathcal{C}(k)$ durant la fenêtre de temps $k \in K$.

— r_i est la capacité résiduelle du capteur $i \in I$

L'objectif de MEC est de minimiser la consommation énergétique totale, ce qui équivaut à maximiser la somme des capacités résiduelles. Pour des raisons de clarté, cette dernière version de l'objectif sera considérée dans le reste du présent chapitre. Enfin, on peut observer que MRC, MCG et MEC partagent le même problème auxiliaire pour l'algorithme de génération de colonnes, celui-ci sera présenté à la section 4.4.2.

La contrainte (4.25) assure la garantie de couverture pour chaque face, d'une valeur de $T^*_{\min}$ unités de temps. MPMCG et MPMEC partagent les mêmes contraintes (4.14)-(4.15)

Modèle MPMEC

$$\min \sum_{i \in I} (E_i - r_i) \Leftrightarrow \max \sum_{i \in I} r_i \tag{4.24}$$

(4.14) − (4.15)

$$\sum_{i \in S(f)} r_i \geq T^*_{\min} \qquad \forall f \in F^* \quad [\lambda_f] \tag{4.25}$$

(4.17), (4.23)

qui imposent la couverture des cibles et le non-dépassement des capacités des capteurs. De même, toutes les couvertures valides pour MPMCG le sont pour MPMEC.

On peut toutefois remarquer que MPMEC ne partage pas le même espace des solutions que MPMCG. En effet, il est possible de construire un ordonnancement réalisable pour MPMCG, mais dont la garantie de couverture est strictement inférieure à $T^*_{\min}$. De manière similaire, les solutions réalisables de MPMRC ne sont pas nécessairement réalisables pour MPMCG, puisque MPMRC tolère les capacités résiduelles négatives. En fait, à chaque étape, l'espace des solutions réalisables est réduit, et il s'ensuit que les solutions réalisables de MPMEC (respectivement MPMCG) sont incluses dans celles de MPMCG (respectivement MPMRC).

4.3.4 Autres formulations pour MCG

Une formulation non-linéaire pour MCG, notée NLMPMCG, est définie par les équations (4.26)-(4.35). Les variables de décision sont les suivantes :

- d_c^k est la durée d'activation de la couverture $c \in \mathcal{C}(k)$ pendant la fenêtre de temps $k \in K$
- r_i est la capacité résiduelle du capteur $i \in I$
- $T_{\min}$ est la garantie de couverture
- x_{ic}^k est égale à 1 si et seulement si le capteur $i \in I$ appartient à la couverture $c \in \mathcal{C}(k)$ dans la fenêtre de temps $k \in K$

La contrainte (4.27) assure que l'énergie consommée par les capteurs ne dépasse pas leur capacité. La contrainte (4.28) assure la surveillance continue des cibles dans chaque fenêtre de temps. La contrainte (4.29) définit $T_{\min}$ comme la garantie de couverture. La contrainte (4.30) garantit qu'au moins un capteur est activé pour surveiller chaque face. La

Modèle NLMPMCG

$$\max T_{\min} \tag{4.26}$$

$$\sum_{k\in K}\sum_{c\in\mathcal{C}(k)} x_{ic}^k d_c^k + r_i = E_i \qquad \forall i \in I \tag{4.27}$$

$$\sum_{c\in\mathcal{C}(k)} d_c^k = \Delta_k \qquad \forall k \in K \tag{4.28}$$

$$\sum_{i\in S(f)} r_i \geq T_{\min} \qquad \forall f \in F^* \tag{4.29}$$

$$\sum_{i\in S(f)} x_{ic}^k \geq 1 \qquad \forall k \in K, \forall c \in \mathcal{C}(k), \forall f \in T(k) \tag{4.30}$$

$$d_c^k \leq d_{c+1}^k \qquad \forall k \in K, \forall c \in \mathcal{C}(k) \setminus \{|\mathcal{C}(k)|\} \tag{4.31}$$

$$T_{\min} \geq 0 \tag{4.32}$$

$$r_i \geq 0 \qquad \forall i \in I \tag{4.33}$$

$$x_{ic}^k \in \{0,1\} \qquad \forall i \in I, \forall k \in K, \forall c \in \mathcal{C}(k) \tag{4.34}$$

$$d_c^k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \tag{4.35}$$

contrainte (4.31) exploite la symmétrie du modèle pour aider le solveur à converger plus rapidement.

À cause de la contrainte (4.30), NLMPMCG est non-linéaire. Nous introduisons une variable continue u_{ic}^k pour remplacer $x_{ic}^k d_c^k$. Le modèle IMPMCG, défini par les équations (4.36)-(4.49), est un programme linéaire en variables mixtes.

La contrainte (4.37) est une version mise à jour de la contrainte (4.30). Les contraintes (4.42) à (4.44) ont été ajoutées pour assurer que u_{ic}^k est égal à $x_{ic}^k d_c^k$.

4.3.5 Bornes supérieures

Dans la présente section, nous présentons des bornes supérieures pour MCG et MEC. Concernant MRC, une borne supérieure évidente est $\text{UB}^{\text{MRC}} = \min_{i\in I} E_i$. Toutefois, cette borne ne nous est pas utile puisque MRC n'a vocation à être résolu de manière optimale que dans le cas où $r_{\min}$ est strictement négatif (on conclut alors que le problème n'a pas de solution). On rappelle que la résolution de MRC est interrompue dès que la non-négativité de $r_{\min}$ est établie, car on dispose alors des colonnes permettant de résoudre MCG.

Modèle IMPMCG

$$\max T_{\min} \tag{4.36}$$

$$\sum_{k\in K}\sum_{c\in\mathcal{C}(k)} u_{ic}^k + r_i = E_i \qquad \forall i \in I \tag{4.37}$$

$$\sum_{c\in\mathcal{C}(k)} d_c^k = \Delta_k \qquad \forall k \in K \tag{4.38}$$

$$\sum_{i\in S(f)} r_i \geq T_{\min} \qquad \forall f \in F^* \tag{4.39}$$

$$\sum_{i\in S(f)} x_{ic}^k \geq 1 \qquad \forall k \in K, \forall c \in \mathcal{C}(k), \forall f \in T(k) \tag{4.40}$$

$$d_c^k \leq d_{c+1}^k \qquad \forall k \in K, \forall c \in \mathcal{C}(k) \setminus \{|\mathcal{C}(k)|\} \tag{4.41}$$

$$u_{ic}^k \leq d_c^k \qquad \forall k \in K, \forall i \in \bigcup_{f\in T(k)} S(f), \forall c \in \mathcal{C}(k) \tag{4.42}$$

$$u_{ic}^k \leq E_i x_{ic}^k \qquad \forall k \in K, \forall i \in \bigcup_{f\in T(k)} S(f), \forall c \in \mathcal{C}(k) \tag{4.43}$$

$$u_{ic}^k \geq \Delta_k(x_{ic}^k - 1) + d_c^k \qquad \forall k \in K, \forall i \in \bigcup_{f\in T(k)} S(f), \forall c \in \mathcal{C}(k) \tag{4.44}$$

$$T_{\min} \geq 0 \tag{4.45}$$

$$r_i \geq 0 \qquad \forall i \in I \tag{4.46}$$

$$u_{ic}^k \geq 0 \qquad \forall i \in I, \forall k \in K, \forall c \in \mathcal{C}(k) \tag{4.47}$$

$$x_{ic}^k \in \{0,1\} \qquad \forall i \in I, \forall k \in K, \forall c \in \mathcal{C}(k) \tag{4.48}$$

$$d_c^k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \tag{4.49}$$

Bornes supérieures pour MCG

Soit $T^*_{\min}$ la garantie de couverture maximale. Celle-ci est bornée par $\mathrm{UB1}^{\mathrm{MCG}}$, l'énergie disponible pour la surveillance de chaque face $f \in F^*$ à laquelle on soustrait la demande de couverture des cibles la traversant.

$$\mathrm{UB1}^{\mathrm{MCG}} = \min_{f \in F^*} \left\{ \sum_{i \in S(f)} E_i - \sum_{k \in K | f \in T(k)} \Delta_k \right\}$$

La borne $\mathrm{UB1}^{\mathrm{MCG}}$ peut être généralisée en considérant des ensembles de faces. Soit $\mathcal{F} \subseteq F^*$ un sous-ensemble des faces chevauchant la zone d'intérêt. La borne $\mathrm{UB2}^{\mathrm{MCG}}$ est calculée de façon similaire, en considérant l'union de faces au lieu d'une seule face.

$$\mathrm{UB2}^{\mathrm{MCG}} = \min_{\mathcal{F} \subseteq F^*} \left\{ \sum_{i \in \bigcup_{f \in \mathcal{F}} S(f)} E_i - \sum_{k \in K | \mathcal{F} \cap T(k) \neq \emptyset} \Delta_k \right\}$$

Bornes supérieures pour MEC

L'énergie consommée par les capteurs ne peut dépasser la somme des capacités des capteurs à laquelle on soustrait l'horizon de temps. Cette borne supérieure est notée $\mathrm{UB1}^{\mathrm{MEC}}$.

$$\mathrm{UB1}^{\mathrm{MEC}} = \sum_{i \in I} E_i - \sum_{k \in K} \Delta_k$$

La borne $\mathrm{UB1}^{\mathrm{MEC}}$ peut être améliorée à l'aide de coefficients α_k, bornes inférieures sur le nombre minimal de capteurs à activer durant une fenêtre de temps k pour couvrir toutes les cibles.

$$\mathrm{UB2}^{\mathrm{MEC}} = \sum_{i \in I} E_i - \sum_{k \in K} \alpha_k \Delta_k$$

La valeur maximale de α_k peut être obtenue en résolvant un problème de couverture d'ensembles. Ceci se faisant au prix d'un temps de calcul élevé, puisqu'il s'agit d'un problème NP-difficile. Cependant, il est possible d'obtenir rapidement une borne inférieure grâce à une solution approchée au problème de clique maximum.

Soit k l'indice d'une fenêtre de temps. Un graphe non-orienté $G = (T(k), E)$ est construit de la manière suivante. À chaque face $f \in T(k)$ correspond un sommet. Une

paire de sommets f et f' est connectée par une arête si et seulement si $S(f) \cap S(f') = \emptyset$, c'est-à-dire si les deux faces ne sont couvertes par aucun capteur commun. Soit $C \subseteq T(k)$ une clique de G. Par définition, au moins $|C|$ capteurs sont nécessaires pour couvrir toutes les faces de C. Par conséquent, $|C|$ est une borne inférieure sur le nombre de capteurs actifs d'une couverture réalisable. Trouver la clique de taille maximale est un problème NP-complet. Toutefois, puisque toute clique fournit une borne inférieure valide, un algorithme approché (Grosso *et al.* 2007) est approprié pour calculer une valeur de α_k, notamment lorsque le nombre de fenêtres de temps est grand.

Bornes supérieures duales pour MCG et MEC

Grâce à la formulation duale des problèmes MCG et MEC, il est possible de calculer des bornes additionnelles. La formulation duale de MPMCG est notée DMPMCG.

Modèle DMPMCG

$$\min \sum_{i \in I} E_i \pi_i + \sum_{k \in K} \Delta_k \mu_k \tag{4.50}$$

$$\pi_i - \sum_{f : i \in S(f)} \lambda_f \geq 0 \qquad \forall i \in I \quad [r_i] \tag{4.51}$$

$$\sum_{f \in F^*} \lambda_f \geq 1 \qquad [T_{\min}] \tag{4.52}$$

$$\sum_{i \in I} a_{ic} \pi_i + \mu_k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \quad [d_c^k] \tag{4.53}$$

$$\pi_i \in \mathbb{R} \qquad \forall i \in I \tag{4.54}$$

$$\mu_k \in \mathbb{R} \qquad \forall k \in K \tag{4.55}$$

$$\lambda_f \leq 0 \qquad \forall f \in F^* \tag{4.56}$$

Considérons un programme linéaire noté RMPMCG, identique à MPMCG, mais composé d'un sous-ensemble réduit des variables de MPMCG. RMPMCG représente ainsi la formulation d'un problème maître restreint pour un algorithme de génération de colonnes. Les variables duales optimales de RMPMCG sont notées $\widetilde{\pi}$, $\widetilde{\mu}$ et $\widetilde{\lambda}$. Nous définissons $|K|$ sous-problèmes notés PP_k :

$$\max_{c \in \mathcal{C}(k)} z_k = -\sum_{i \in I} \widetilde{\pi}_i a_{ic} - \widetilde{\mu}_k$$

Soit z_k^* la valeur optimale de la fonction objectif de PP_k. Par définition, $z_k^* \geq -\sum_{i\in I} \widetilde{\pi}_i a_{ic} - \widetilde{\mu}_k$ pour tout $k \in K$, et pour tout $c \in \mathcal{C}(k)$. Ceci peut être réécrit $\sum_{i\in I} \widetilde{\pi}_i a_{ic} + (\widetilde{\mu}_k + z_k^*) > 0$ pour tout $k \in K$, et pour tout $c \in \mathcal{C}(k)$.

Affecter :

$$\begin{bmatrix} \pi \\ \mu \\ \lambda \end{bmatrix} = \begin{bmatrix} \widetilde{\pi} \\ \widetilde{\mu} + z^* \\ \widetilde{\lambda} \end{bmatrix}$$

produit une solution réalisable pour DMPMCG. En effet, la contrainte (4.53) est satisfaite par définition de $z^* = \left[z_1^* \dots z_{|K|}^*\right]^\top$, les contraintes (4.51) et (4.52) sont satisfaites par $\widetilde{\pi}$ et $\widetilde{\lambda}$ parce que toutes les variables r_i et $T_{\min}$ sont présentes dans RMPMCG, donc leurs contraintes correspondantes sont présentes dans le dual de RMPMCG.

Grâce à la propriété de la dualité faible, l'objectif de toute solution réalisable de DMPMCG est une borne supérieure sur l'objectif de MPMCG. Par conséquent :

$$T_{\min} \leq \sum_{i\in I} E_i \widetilde{\pi}_i + \sum_{k\in K} \Delta_k (\widetilde{\mu}_k + z_k^*)$$

Par un raisonnement analogue, on peut obtenir une borne pour MEC. La formulation duale de MPMEC est notée DMPMEC.

Modèle DMPMEC

$$\min \sum_{i\in I} E_i \pi_i + \sum_{k\in K} \Delta_k \mu_k + T_{\min}^* \sum_{f\in F^*} \lambda_f \tag{4.57}$$

$$\pi_i - \sum_{f: i\in S(f)} \lambda_f \geq 1 \qquad \forall i \in I \quad [r_i] \tag{4.58}$$

$$\sum_{i\in I} a_{ic}\pi_i + \mu_k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \quad [d_c^k] \tag{4.59}$$

$$\pi_i \in \mathbb{R} \qquad \forall i \in I \tag{4.60}$$

$$\mu_k \in \mathbb{R} \qquad \forall k \in K \tag{4.61}$$

$$\lambda_f \leq 0 \qquad \forall f \in F^* \tag{4.62}$$

On considère alors RMPMEC, identique à MPMEC, mais composé d'un sous-ensemble des variables de MPMEC. Les variables duales optimales de RMPMEC sont notées $\widetilde{\pi}$, $\widetilde{\mu}$ et

$\widetilde{\lambda}$. Nous définissons $|K|$ sous-problèmes notés PP_k :

$$\max_{c \in \mathcal{C}(k)} z_k = -\sum_{i \in I} \widetilde{\pi}_i a_{ic} - \widetilde{\mu}_k$$

Soit z_k^* la valeur optimale de la fonction objectif de PP_k. Par définition, $z_k^* \geq -\sum_{i \in I} \widetilde{\pi}_i a_{ic} - \widetilde{\mu}_k$ pour tout $k \in K$, et pour tout $c \in \mathcal{C}(k)$. Ceci peut être réécrit $\sum_{i \in I} \widetilde{\pi}_i a_{ic} + (\widetilde{\mu}_k + z_k^*) > 0$ pour tout $k \in K$, et pour tout $c \in \mathcal{C}(k)$.

Affecter :

$$\begin{bmatrix} \pi \\ \mu \\ \lambda \end{bmatrix} = \begin{bmatrix} \widetilde{\pi} \\ \widetilde{\mu} + z^* \\ \widetilde{\lambda} \end{bmatrix}$$

produit une solution réalisable pour DMPMEC. En effet, la contrainte (4.59) est satisfaite par définition de $z^* = \begin{bmatrix} z_1^* \dots z_{|K|}^* \end{bmatrix}^\top$, la contrainte (4.58) est satisfaite par $\widetilde{\pi}$ et $\widetilde{\lambda}$ parce que toutes les variables r_i sont présentes dans RMPMEC, donc leurs contraintes correspondantes sont présentes dans le dual de RMPMEC.

Grâce à la propriété de la dualité faible, l'objectif de toute solution réalisable de DMPMEC est une borne supérieure sur l'objectif de MPMEC. Par conséquent :

$$\sum_{i \in I} r_i \leq \sum_{i \in I} E_i \widetilde{\pi}_i + \sum_{k \in K} \Delta_k (\widetilde{\mu}_k + z_k^*)$$

4.4 Résolution

Pour mémoire, le problème d'ordonnancement d'activités de capteurs de ce chapitre se décompose en trois sous-problèmes MRC, MCG et MEC. À l'issue de la discrétisation, la méthode proposée se déroule ainsi en trois étapes correspondant aux trois sous-problèmes (figure 4.2). La présente section introduit la méthode de résolution commune aux trois sous-problèmes.

FIGURE 4.2 – Les trois étapes de la méthode

4.4.1 Algorithme de génération de colonnes

Puisque le nombre de couvertures valides pour ces sous-problèmes est en $\mathcal{O}(|K|2^m)$, il est en pratique impossible de les énumérer explicitement en un temps raisonnable. Par conséquent, résoudre directement leurs modèles MPMRC, MPMCG et MPMEC à l'aide d'un solveur n'est pas envisageable.

C'est pour cela qu'une méthode à base de génération de colonnes est appropriée. Son idée principale est que seul un sous-ensemble des couvertures est requis pour résoudre les programmes linéaires, puisque la plupart des variables dans la solution optimale ne seront pas dans la base. Ainsi, MPMRC, MPMCG et MPMEC sont formulés comme le problème maître d'un algorithme de génération de colonnes. De plus, ils sont formulés de manière à partager le même problème auxiliaire, noté AUX. Par conséquent, chaque sous-problème est résolu de la même manière. L'algorithme de génération de colonnes est illustré par le diagramme de la figure 4.3 et expliqué plus en détail dans la présente section.

Sans perte de généralité, nous utilisons la notation MP pour désigner indifféremment l'un des trois programmes linéaires MPMRC, MPMCG ou MPMEC, incluant toutes les couvertures. La génération de colonnes débute avec RMP, pour *Restricted Master Problem*, un programme linéaire identique à MP, à ceci près qu'il ne contient qu'un petit sous-ensemble de couvertures de MP. Le sous-ensemble de couvertures initial qui constitue les colonnes de RMP est choisi arbitrairement, de manière à ce que RMP soit réalisable. Le problème auxiliaire, noté AUX, consiste à construire des couvertures profitables pour RMP, c'est-à-dire trouver des variables de MP dont le coût réduit est positif. Il s'ensuit que l'ajout de telles couvertures dans RMP améliore la valeur de sa fonction objectif. Il n'est toutefois pas nécessaire de résoudre AUX de manière exacte, sauf pour prouver l'optimalité de la solution de MP, puisqu'une couverture de coût réduit positif est suffisante. Ainsi, pour gagner en temps de calcul, AUX peut être résolu par une métaheuristique. L'algorithme de génération de colonnes est arrêté dès lors qu'il n'existe plus de couverture profitable.

Dans la présente section, nous présentons le problème auxiliaire puis la manière de générer des couvertures initiales.

4.4.2 Problème auxiliaire

Le problème auxiliaire AUX est décomposable en $|K|$ sous-problèmes indépendants notés AUX_k et formulés à l'aide de programmes linéaires en variables binaires. Pour toute fenêtre de temps $k \in K$, le programme linéaire en variables binaires formulant AUX_k est

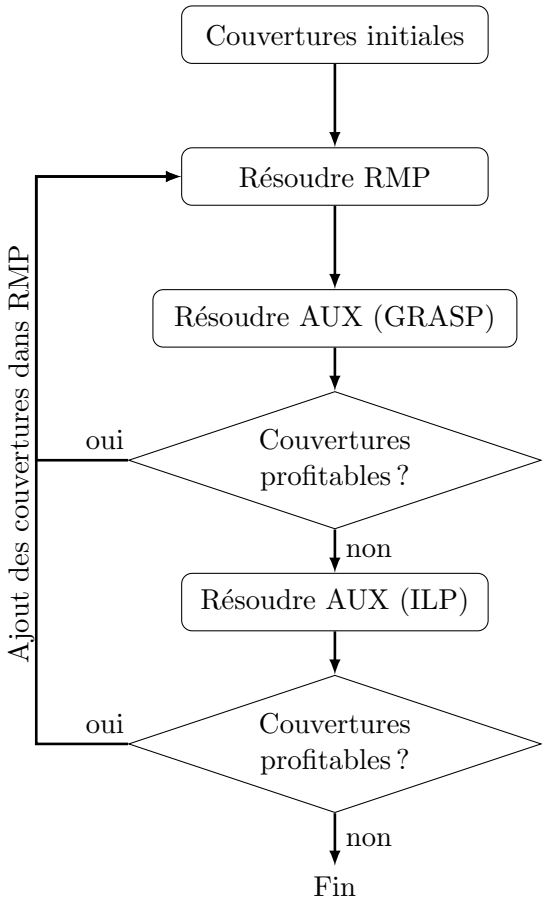


FIGURE 4.3 – Algorithme de génération de colonnes

défini par les équations (4.63)-(4.66). Les coefficients π_i et μ_k sont les valeurs des variables duales quand RMP a été résolu de manière optimale. La variable a_{ic} vaut 1 si le capteur i est sélectionné dans la nouvelle couverture c, 0 sinon.

Modèle AUX_k

$$\max \; -\sum_{i \in I} \pi_i a_{ic} - \mu_k \tag{4.63}$$

$$\sum_{i \in S(f)} a_{ic} \geq 1 \qquad \forall f \in T(k) \tag{4.64}$$

$$\sum_{i \in I} a_{ic} \leq |T(k)| \tag{4.65}$$

$$a_{ic} \in \{0, 1\} \qquad \forall i \in I \tag{4.66}$$

L'objectif de AUX_k est de maximiser le coût réduit de la nouvelle variable d_c^k, afin de construire une couverture profitable. La contrainte (4.64) assure la couverture de toutes les faces de $f \in T(k)$, en sélectionnant au moins un capteur candidat parmi $S(f)$. La contrainte (4.65) borne le nombre de capteurs à sélectionner pour réduire l'espace des solutions. En effet, il n'est pas requis d'activer plus de capteurs que de faces à couvrir.

Trouver une couverture de coût réduit positif est suffisant pour améliorer la fonction objectif de RMP. Par conséquent, il n'est pas nécessaire de calculer la solution optimale de AUX_k, sauf pour démontrer l'optimalité de la solution courante lorsqu'il n'existe plus de couverture de coût réduit positif. Ainsi, la génération de colonnes peut être accélérée à l'aide d'une métaheuristique comme GRASP (Feo *et al.* 1995). GRASP, décrit dans l'algorithme 5, construit une couverture itérativement. Cette métaheuristique est paramétrable via une valeur $\alpha \in [0, 1]$. Une valeur de α plus proche de 0 signifie que le comportement de l'algorithme sera davantage aléatoire, tandis qu'avec une valeur proche de 1, l'algorithme sera davantage déterministe. À chaque itération, GRASP calcule pour chaque capteur une utilité $u_i = \frac{n_i}{\pi_i}$, où n_i est le nombre de faces non encore couvertes que le capteur i peut couvrir. Le seuil $u_{\text{limit}} = u_{\min} + \alpha(u_{\max} - u_{\min})$ permet de sélectionner les capteurs ayant les utilités les plus élevées. Un capteur dont l'utilité est supérieure ou égale à u_{limit} est sélectionné aléatoirement et ajouté à la couverture. Ainsi, l'algorithme sélectionne en priorité les capteurs qui couvrent le plus grand nombre de cibles (n_i) à moindre coût (π_i). Ce processus est répété jusqu'à ce que la couverture soit réalisable, c'est-à-dire que toutes les faces sont couvertes par un capteur actif.

Algorithme 5 : GRASP

Input : $\alpha \in [0,1]$ paramètre de GRASP
$\forall i \in I, \quad a_{ic} \leftarrow 0$ // Ensemble des capteurs sélectionnés
$\mathcal{F} \leftarrow T(k)$ // Ensemble des faces non-encore couvertes
$\mathcal{F}_i \leftarrow \{f \in \mathcal{F} | i \in S(f)\}$ // Faces non-encore couvertes à la portée du capteur i
// Tant qu'il reste des faces non-couvertes
while $|\mathcal{F}| > 0$ **do**
  // Calcul des utilités
  $\forall i \in I, \quad u_i \leftarrow \frac{|\mathcal{F}_i|}{\pi_i}$
  // Calcul du seuil u_{limit}
  $u_{\min} \leftarrow \min_{i \in I} u_i$
  $u_{\max} \leftarrow \max_{i \in I} u_i$
  $u_{\text{limit}} \leftarrow u_{\min} + \alpha(u_{\max} - u_{\min})$
  // Création de la liste des candidats RCL
  $\text{RCL} \leftarrow \{i \in I | u_i \geq u_{\text{limit}}\}$
  // Sélection aléatoire d'un capteur de RCL
  $\widehat{i} \leftarrow$ `ChooseRandom(`RCL`)`
  // Mise à jour des données
  $a_{\widehat{i}c} \leftarrow 1$
  $\mathcal{F} \leftarrow \mathcal{F} \setminus \{f \in F | \widehat{i} \in S(f)\}$
  $\mathcal{F}_i \leftarrow \mathcal{F}_i \setminus \{f \in F | \widehat{i} \in S(f)\}, \forall i \in I$
return $(a_{1c}, \ldots, a_{mc})$

Afin d'accroître les chances d'obtenir une couverture de coût réduit positif, GRASP peut être couplé à un algorithme de recherche locale. La recherche locale est un algorithme d'exploration qui énumère le voisinage de la solution retournée par le GRASP dans le but de l'améliorer. Pour cela, on effectue des échanges de capteurs susceptibles d'augmenter la valeur du coût réduit. Plusieurs voisinages sont testés successivement, par ordre croissant de leur taille. Tout d'abord, on tente de supprimer des capteurs, autrement dit, on cherche à effectuer des 1-0-échanges. Puis on tente d'échanger un capteur avec un autre plus profitable, en effectuant un 1-1-échange. L'échange de deux capteurs contre un autre est un 2-1-échange. Ceci peut être étendu au a-b-échange où a est le nombre de capteurs à supprimer et b le nombre de capteurs à ajouter. Toutefois, plus les valeurs de a et b augmentent, plus le voisinage est vaste à cause du nombre de combinaisons possibles. Ceci a des répercussions sur les temps de calcul et peut rendre la recherche locale improductive. Par conséquent, nous suggérons de limiter la recherche locale au 2-1-échange. Chaque fois qu'un échange améliorant le coût réduit est trouvé, il est immédiatement effectué, et la recherche est poursuivie jusqu'à ce que toutes les possibilités d'échanges aient été essayées.

En résumé, le problème auxiliaire est résolu à l'aide de GRASP et d'une recherche locale, tant que ces derniers trouvent des couvertures de coût réduit positif. Le cas échéant, la génération de colonnes fait appel à un solveur de programmes linéaires en variables binaires, afin de trouver de nouvelles couvertures, ou prouver l'optimalité de la solution courante.

4.4.3 Couvertures initiales

La contrainte (4.15) des modèles MPMRC, MPMCG et MPMEC ne peut être satisfaite sans un ensemble de couvertures initiales, en l'absence de membres de gauche dans l'équation. Par conséquent, il est nécessaire de générer un ensemble de couvertures initiales pour commencer l'algorithme de génération de colonnes car d'après le lemme 6, déterminer si MRC est réalisable est un problème NP-complet.

De plus, nous n'avons pas de valeurs de variables duales (π, μ) dont dépendent les problèmes auxiliaires AUX_k. Pour cela, nous choisissons de fixer les valeurs $\pi_i = 1$ et $\mu_k = -|T(k)|-1$ afin de réduire AUX_k à un problème de couverture d'ensembles à coûts unitaires. Ce choix permet de générer en premier lieu des couvertures de cardinalité minimale, qui sont souhaitables pour la résolution de MEC.

Toutefois, la génération de l'ensemble de couvertures initiales n'est pas suffisante pour obtenir un ordonnancement réalisable. Par exemple, il est possible de générer une couver-

ture activant tous les capteurs simultanément, et de l'ajouter à toutes les fenêtres de temps. Néanmoins, une telle couverture est très coûteuse en énergie. À moins que les capteurs aient une capacité suffisamment grande, il n'existe pas d'ordonnancement réalisable respectant les capacités. Ceci justifie le rôle du sous-problème MRC, qui autorise temporairement des capacités résiduelles r_i négatives dans le modèle MPMRC. Initialement, l'objectif de MPMRC est généralement négatif et augmente au fur et à mesure de la génération de colonnes. L'ensemble de couvertures devient réalisable dès lors qu'il existe un ordonnancement tel que les capacités résiduelles r_i sont positives ou nulles, c'est-à-dire dès lors que la valeur optimale de $r_{\min}$ (dans MPMRC) est positive ou nulle.

4.4.4 Exemple

Résolvons l'instance décrite à la section 2.6 (page 28), qui comporte 3 capteurs et 2 cibles. Les capacités initiales des capteurs 1, 2 et 3 sont respectivement fixées à $E_1 = 40$, $E_2 = 30$ et $E_3 = 30$. À l'issue de la discrétisation et de la procédure de réduction, nous obtenons une instance à 3 fenêtres de temps dans laquelle 4 faces sont traversées par les cibles.

$$
\begin{array}{lll}
S(1) = \{s_1\} & T(1) = \{f_1\} & \Delta_1 = 30.2948 \\
S(2) = \{s_1, s_3\} & T(2) = \{f_2\} & \Delta_2 = 17.5427 \\
S(3) = \{s_1, s_2, s_3\} & T(3) = \{f_4\} & \Delta_3 = 12.1625 \\
S(4) = \{s_3\} &&
\end{array}
$$

Les sous-problèmes auxiliaires AUX_k sont initialisés avec $\pi_i = 1, \forall i \in I$ et $\mu_k = -2, \forall k \in K$. Après résolution de chacun d'entre eux à l'aide de la métaheuristique GRASP, nous obtenons pour les fenêtres de temps 1, 2 et 3, les couvertures respectives $\{s_1\}$, $\{s_1\}$ et $\{s_3\}$. Ces couvertures sont ajoutées aux programme linéaire MPMRC, qui, une fois résolu à l'aide d'un simplexe, donne une valeur de fonction objectif $r_{\min} = -7.83754$. Les variables duales associées aux contraintes (4.14) sont $\pi_1 = 1$, $\pi_2 = 0$ et $\pi_3 = 0$, celles associées aux contraintes (4.15) sont $\mu_1 = -1$, $\mu_2 = -1$ et $\mu_3 = 0$. Après résolution des sous-problèmes AUX_k avec les nouveaux coefficients, seule une couverture de coût réduit strictement positif est générée, $\{s_3\}$ dans la fenêtre de temps 2. Une nouvelle résolution de MPMRC retourne $r_{\min} = 5$. Par conséquent, l'ensemble de couvertures courant est réalisable pour MPMCG et la résolution du sous-problème MCG peut débuter. La résolution de MPMCG renvoie les

variables duales suivantes : $\pi_1 = \frac{1}{2}$, $\pi_2 = 0$, $\pi_3 = \frac{1}{2}$, $\mu_1 = -\frac{1}{2}$, $\mu_2 = -\frac{1}{2}$ et $\mu_3 = -\frac{1}{2}$. Il n'est pas possible d'obtenir de couverture de coût réduit strictement positif, alors l'algorithme passe à la dernière phase (MEC). La garantie de couverture optimale est $T^*_{\min} = 5$. À l'issue de la résolution de MPMEC, on obtient les variables duales $\pi_1 = 1$, $\pi_2 = 1$, $\pi_3 = 1$, $\mu_1 = -1$, $\mu_2 = -1$ et $\mu_3 = -1$. De même, il n'est pas possible de générer de couvertures de coût réduit strictement positif. Ainsi, un ordonnancement optimal peut être construit, dans lequel le capteur 1 est activé durant l'intervalle $[0, 35]$ et le capteur 2 durant l'intervalle $[35, 60]$. Le réseau de capteurs consomme au total 60 unités d'énergie.

4.5 Résultats numériques

Les tests numériques sont effectués sur un jeu de 100 instances générées aléatoirement. Dans chacune d'elles, les trajectoires des n cibles ($n \in \{5, 10, 15\}$) sont décrites à l'intérieur d'une région carrée de dimensions 100×100. Les trajectoires sont représentées par une ligne polygonale composée de 4 segments reliés par 5 points de contrôle. À chaque point de contrôle est associé un temps de passage de la cible correspondante. Les temps de passage associés aux points de contrôle sont répartis uniformément le long de l'horizon de temps qui vaut $H = 100$ unités de temps. Autrement dit, le premier point de contrôle d'une cible correspond à sa position à $t = 0$, le second à $t = 25$, le troisième à $t = 50$, etc. Les temps de passage entre chaque paire de points de contrôle consécutifs sont uniformémement distribués, c'est-à-dire que la vitesse des cibles est constante jusqu'au prochain point de contrôle. Les points de contrôle sont générés aléatoirement. Il s'ensuit que les cibles sont susceptibles de traverser plusieurs fois une même face. Un ensemble de m capteurs ($m \in \{50, 100, 150\}$) est disséminé aléatoirement, en s'assurant que la zone de couverture de chaque capteur chevauche une partie d'au moins l'une des trajectoires des cibles. Les zones de couverture sont des disques de rayon $R^S = 40$. Les capacités initiales des capteurs sont aléatoires, variant dans l'intervalle $[0, 100]$. En résumé, pour chaque jeu de paramètres m et n, 10 instances sont créées, on obtient alors 3×3 groupes de 10 instances. En outre, on ajoute un jeu de 10 instances additionnelles avec $m = 50$ capteurs et $n = 30$ cibles. On dénombre alors au total 100 instances.

L'implémentation du programme est réalisée en C++ et exécutée sur une machine équipée d'un processeur Intel Xeon W3520 (2.67 GHz $\times$ 8) avec 8 Go de RAM sous Linux Ubuntu 14.04. Les programmes linéaires MPMRC, MPMCG et MPMEC sont résolus à l'aide de CPLEX 12.6.1. Les problèmes auxiliaires AUX_k sont résolus de manière approchée

à l'aide d'une implémentation de GRASP et de manière exacte à l'aide de CPLEX. Pour GRASP, plusieurs valeurs du paramètre de randomisation α ont été testées, dans l'ensemble $\{0.6, 0.7, 0.8, 0.9, 1\}$. Chaque variante et jeu de paramètres a été exécuté 10 fois sur chaque instance.

m	n	Moy. $\lvert K\rvert$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$	$\alpha = 1$
50	5	515	0.576 s	0.602 s	**0.566** s	0.572 s	0.589 s
	10	936	1.419 s	1.420 s	**1.411** s	1.449 s	1.478 s
	15	1304	2.455 s	2.456 s	**2.301** s	2.506 s	2.490 s
	30	1955	5.623 s	5.400 s	**5.189** s	5.483 s	5.699 s
100	5	1080	1.468 s	1.445 s	1.486 s	**1.378** s	1.428 s
	10	2100	4.259 s	**3.990** s	4.059 s	4.247 s	4.245 s
	15	3061	8.614 s	8.427 s	8.115 s	**8.058** s	8.003 s
150	5	1640	**2.793** s	3.059 s	3.023 s	2.944 s	2.930 s
	10	3260	10.05 s	**9.042** s	9.234 s	11.17 s	10.55 s
	15	4797	18.93 s	18.69 s	**17.96** s	19.11 s	19.10 s

TABLEAU 4.3 – Moyenne du temps d'exécution en fonction de la valeur α de GRASP

Dans un premier temps, on considère que les positions des cibles ne sont pas incertaines (rayon du disque d'incertitude $\delta = 0$). Le tableau 4.3 compare les temps de calcul moyens en fonction du paramètre de randomisation α de GRASP. Ces temps ne tiennent pas compte du temps écoulé lors de la discrétisation. Bien que l'impact du coefficient α soit faible, il semble que les meilleurs temps de calcul se situent autour de $\alpha = 0.8$. Ce faible impact s'explique par le fait que les sous-problèmes auxiliaires sont de petite taille, et que la recherche locale est particulièrement efficace sur ceux-ci. Dans la suite de cette section, nous choisissons $\alpha = 0.8$ pour le paramètre de randomisation de GRASP.

Le tableau 4.4 compare les temps de calcul de l'approche matheuristique (combinant la métaheuristique GRASP et le solveur CPLEX) notée GRASP+ILP, avec les temps de la génération de colonnes basique, qui résout tous les problèmes AUX_k uniquement à l'aide de CPLEX (ILP). La troisième colonne contient le nombre moyen de fenêtres de temps (après réduction de la taille de l'instance).

La matheuristique GRASP+ILP améliore le temps de calcul moyen d'un facteur 3 à 4 par rapport à l'approche ILP. Par ailleurs, on remarque que la difficulté à résoudre le problème augmente sensiblement avec la taille de l'instance, en particulier lorsque le nombre de fenêtres de temps augmente. En effet, le nombre de fenêtres de temps détermine le nombre de sous-problèmes auxiliaires à résoudre à chaque itération de l'algorithme de

m	n	Moy. $\lvert K\rvert$	GRASP+ILP	ILP
50	5	515	0.566 s	1.938 s
	10	936	1.411 s	4.749 s
	15	1304	2.301 s	7.926 s
	30	1955	5.189 s	16.30 s
100	5	1080	1.486 s	5.191 s
	10	2100	4.059 s	15.70 s
	15	3061	8.115 s	28.29 s
150	5	1640	3.023 s	11.52 s
	10	3260	9.234 s	29.20 s
	15	4797	17.96 s	68.88 s

TABLEAU 4.4 – Moyenne du temps d'exécution de GRASP+ILP et ILP en fonction du nombre de capteurs

génération de colonnes.

m	n	$\delta = 0$	$\delta = 1$	$\delta = 2$
50	5	515	598	607
	10	936	1163	1210
	15	1304	1677	1760
	30	1955	2779	2938
100	5	1080	1295	1619
	10	2100	2687	3379
	15	3061	4072	5114
150	5	1640	2181	2769
	10	3260	4646	5787
	15	4797	7107	8781

TABLEAU 4.5 – Moyenne du nombre $\lvert K\rvert$ de fenêtres de temps en fonction du rayon δ du disque d'incertitude

Ajouter de l'incertitude spatiale modifie le nombre de fenêtres de temps générées à l'issue de la discrétisation. En effet, un disque d'incertitude est susceptible de traverser davantage de frontières de zones de couverture qu'une cible ponctuelle. Le tableau 4.5 montre l'impact du rayon du disque d'incertitude sur le nombre de fenêtres de temps. Un rayon d'incertitude $\delta = 0$ signifie qu'on connaît la position des cibles avec certitude. Ces données portent sur les instances réduites. Le rayon d'incertitude a également un impact sur les temps d'exécution de l'algorithme de discrétisation, comme le montre le tableau 4.6.

m	n	$\delta = 0$	$\delta = 1$	$\delta = 2$
50	5	0.023 s	0.279 s	0.638 s
	10	0.076 s	1.080 s	2.557 s
	15	0.162 s	2.436 s	5.779 s
	30	0.623 s	9.899 s	23.54 s
100	5	0.089 s	2.814 s	7.960 s
	10	0.326 s	11.84 s	34.17 s
	15	0.718 s	27.58 s	79.31 s
150	5	0.200 s	11.72 s	37.50 s
	10	0.761 s	52.15 s	169.0 s
	15	1.691 s	124.6 s	396.2 s

TABLEAU 4.6 – Moyenne du temps d'exécution de la discrétisation en fonction du rayon δ du disque d'incertitude

La procédure de réduction joue un rôle important dans le cas de l'incertitude. Le tableau 4.7 montre l'efficacité de cette procédure en fonction du nombre de capteurs, de cibles, et du rayon du disque d'incertitude. Dans le cas sans incertitude, on observe que le nombre de fenêtres supprimées augmente sensiblement en fonction du nombre de cibles. En effet, avec un plus grand nombre de cibles, il y a de plus fortes chances d'obtenir des faces redondantes à couvrir, c'est-à-dire des faces dont l'ensemble de capteurs candidats forme un sur-ensemble des capteurs candidats d'une autre face. Ceci offre davantage d'opportunités de fusionner des fenêtres de temps.

m	n	$\delta = 0$	$\delta = 1$	$\delta = 2$
50	5	7 %	46 %	45 %
	10	16 %	48 %	46 %
	15	22 %	50 %	47 %
	30	41 %	58 %	55 %
100	5	3 %	42 %	27 %
	10	7 %	41 %	25 %
	15	10 %	40 %	25 %
150	5	2 %	35 %	17 %
	10	4 %	32 %	14 %
	15	6 %	30 %	14 %

TABLEAU 4.7 – Pourcentage moyen de fenêtres de temps supprimées après réduction des instances en fonction du nombre de capteurs, de cibles et du rayon δ du disque d'incertitude

Lorsque le rayon du disque d'incertitude est égal à 1, entre de 30% et 60% des fenêtres sont supprimées en moyenne. Le potentiel de la procédure de réduction est augmenté lorsqu'on considère un disque d'incertitude, puisque le nombre de fenêtres de temps est généralement sensiblement plus élevé. Toutefois ce phénomène a des limites, comme le montre le cas $\delta = 2$. En effet, il est possible de produire moins de fenêtres de temps avec un disque d'incertitude plus grand.

Par exemple, prenons le cas particulier d'une cible dont la trajectoire est une sinusoïde longeant le bord de la zone de couverture d'un capteur i. Si le rayon d'incertitude est suffisamment petit, la cible traverse deux fois le bord à chaque période. Ainsi, chaque entrée ou sortie de la zone de couverture du capteur i produit un tick. Si on augmente suffisamment son rayon, il s'ensuit que le disque d'incertitude couvre constamment le bord du capteur i, sans jamais en sortir. Par conséquent, le nombre de ticks est diminué, ce qui diminue également le potentiel de la procédure de réduction. Sur nos instances, le nombre de fenêtres de temps est généralement très légèrement inférieur avec $\delta = 2$ par rapport à $\delta = 1$.

m	n	$\delta = 0$	$\delta = 1$	$\delta = 2$
50	5	0.004 s	0.076 s	0.113 s
	10	0.048 s	0.546 s	0.784 s
	15	0.198 s	1.837 s	2.638 s
	30	2.164 s	20.76 s	28.45 s
100	5	0.011 s	0.440 s	0.669 s
	10	0.114 s	3.493 s	5.619 s
	15	0.459 s	16.44 s	21.66 s
150	5	0.019 s	1.204 s	2.306 s
	10	0.176 s	12.43 s	18.54 s
	15	0.686 s	46.66 s	57.72 s

TABLEAU 4.8 – Moyenne du temps d'exécution de la réduction d'instances en fonction du rayon δ du disque d'incertitude

Au fur et à mesure qu'on augmente son rayon, de plus en plus de variétés de faces peuvent être chevauchées par le disque d'incertitude. Nous avons observé que le nombre de faces supprimées dans les ensembles $T(k)$ est significativement plus grand pour $\delta = 2$ que pour $\delta = 1$. Toutefois, le nombre de fenêtres de temps fusionnées est plus petit pour $\delta = 2$. L'augmentation du rayon d'incertitude peut ainsi rendre la procédure de réduction moins efficace.

		$\delta = 0$			$\delta = 1$			$\delta = 2$		
m	n	S.R.	A.R.	Gain	S.R.	A.R.	Gain	S.R.	A.R.	Gain
50	5	0.645 s	0.570 s	12 %	1.303 s	0.693 s	47 %	1.457 s	0.806 s	45 %
	10	1.737 s	1.459 s	16 %	3.886 s	2.195 s	44 %	5.213 s	2.850 s	45 %
	15	3.364 s	2.499 s	26 %	7.821 s	4.936 s	37 %	10.88 s	6.354 s	42 %
	30	10.92 s	7.353 s	33 %	26.43 s	27.73 s	-5 %	38.40 s	37.14 s	3 %
100	5	1.463 s	1.497 s	1 %	3.882 s	2.339 s	40 %	6.763 s	3.500 s	48 %
	10	4.319 s	4.173 s	3 %	13.92 s	9.048 s	35 %	27.81 s	14.65 s	47 %
	15	9.212 s	8.574 s	7 %	37.46 s	28.91 s	23 %	68.04 s	40.95 s	40 %
150	5	3.099 s	3.042 s	2 %	10.84 s	5.813 s	46 %	19.55 s	9.453 s	52 %
	10	9.753 s	9.410 s	4 %	45.38 s	27.88 s	39 %	117.2 s	44.82 s	62 %
	15	18.93 s	18.65 s	2 %	139.0 s	81.32 s	41 %	N/A	114.0 s	N/A

TABLEAU 4.9 – Temps d'exécution de la génération de colonnes, avec ou sans réduction

Le coût en temps de calcul de la réduction d'instances est représenté dans le tableau 4.8. Lorsque la position des cibles est incertaine, le temps d'exécution de la procédure de réduction peut atteindre jusqu'à une minute en moyenne. Toutefois, la réduction d'instances est le plus souvent bénéfique, lorsqu'on compare les temps avec le tableau 4.9, qui montre les temps de calcul de la génération de colonnes avec et sans réduction. Les colonnes nommées S.R. représentent les temps d'exécution de la génération de colonnes sur les instances non-réduites. Les colonnes nommées A.R. représentent ces mêmes temps, sur les instances réduites, auxquels on a additionné le temps d'exécution de la réduction d'instance. Les colonnes "Gain" montrent le pourcentage de temps total procuré par l'utilisation de la procédure de réduction. Pour le cas $\delta = 2$ avec $m = 150$ capteurs et $n = 15$ cibles, sans réduction, l'algorithme a été stoppé avant la fin car le temps de calcul était excessivement élevé. Les gains les plus élevés sont généralement observés pour les cas où les positions de cibles sont incertaines et que le nombre de cibles est inférieur à 15.

Étudions la qualité des bornes supérieures pour chaque valeur de $\delta \in \{0, 1, 2\}$. Le tableau 4.10 montre l'écart moyen en pourcentage entre les bornes supérieures et les valeurs optimales des fonctions objectifs des sous-problèmes MCG et MEC. Le calcul de la borne UB2^{MEC} nécessite la résolution (approchée ou exacte) d'un problème de clique maximum. Pour cela, nous utilisons la bibliothèque LEMON (DEZSŐ *et al.* 2011), et en particulier l'implémentation d'un algorithme de recherche locale itérée proposé par Grosso, Locatelli, and Pullan (GROSSO *et al.* 2007). La borne UB1^{MCG} atteint l'optimalité pour 94 % des instances. Pour rappel, UB1^{MCG} calcule les différences entre le potentiel énergétique des

		$\delta = 0$			$\delta = 1$			$\delta = 2$		
m	n	$UB1^{MCG}$	$UB1^{MEC}$	$UB2^{MEC}$	$UB1^{MCG}$	$UB1^{MEC}$	$UB2^{MEC}$	$UB1^{MCG}$	$UB1^{MEC}$	$UB2^{MEC}$
50	5	< 0.01 %	2.18 %	0.28 %	0 %	2.18 %	0.94 %	< 0.01 %	2.18 %	0.19 %
	10	0 %	3.67 %	0.39 %	0 %	3.67 %	1.21 %	< 0.01 %	3.67 %	0.29 %
	15	0 %	5.10 %	0.51 %	< 0.01 %	5.10 %	1.23 %	< 0.01 %	5.10 %	0.31 %
	30	0 %	7.78 %	0.85 %	0 %	7.78 %	1.46 %	0 %	7.78 %	0.33 %
100	5	0 %	0.90 %	0.11 %	0 %	0.90 %	0.15 %	0 %	0.90 %	0.02 %
	10	0 %	1.53 %	0.13 %	0 %	1.53 %	0.15 %	0 %	1.53 %	0.05 %
	15	0 %	2.23 %	0.22 %	0 %	2.23 %	0.20 %	0 %	2.23 %	0.06 %
150	5	0 %	0.54 %	0.06 %	0 %	0.54 %	0.01 %	0 %	0.54 %	0.01 %
	10	0 %	0.96 %	0.09 %	0 %	0.96 %	0.04 %	0 %	0.96 %	0.04 %
	15	0 %	1.40 %	0.15 %	0 %	1.40 %	0.10 %	0 %	1.40 %	0.08 %

TABLEAU 4.10 – Écart moyen entre les bornes supérieures et les valeurs optimales

faces (somme des capacités résiduelles des capteurs candidats) et la demande de couverture des cibles qui la traversent. Dans tous les cas, la borne $UB2^{MEC}$ domine $UB1^{MEC}$, puisque $UB2^{MEC}$ est une version améliorée de $UB1^{MEC}$. Les bornes sur le problème de clique maximum sont très bénéfiques sur les performances de $UB2^{MEC}$. Ces bornes, en particulier pour MEC, peuvent être utiles dans le cas d'instances difficiles pour lesquelles l'algorithme de génération de colonnes ne retourne pas de solution optimale en temps raisonnable. L'utilisateur peut ainsi juger de la qualité de la solution courante, et décider d'interrompre l'algorithme si la qualité de la solution lui paraît suffisante.

	$m = 20$		$m = 30$		$m = 40$	
Instance	IMPMCG	G+ILP	IMPMCG	G+ILP	IMPMCG	G+ILP
1	1.27 s	0.13 s	3.96 s	0.21 s	> 3600 s	0.29 s
2	1.28 s	0.09 s	4.34 s	0.15 s	10.0 s	0.24 s
3	> 3600 s	0.14 s	> 3600 s	0.08 s	> 3600 s	0.18 s
4	0.86 s	0.11 s	2.50 s	0.18 s	> 3600 s	0.24 s
5	0.92 s	0.07 s	> 3600 s	0.14 s	59.8 s	0.19 s
6	0.49 s	0.10 s	> 3600 s	0.27 s	5.72 s	0.23 s
7	0.80 s	0.09 s	> 3600 s	0.19 s	8.15 s	0.29 s
8	0.68 s	0.11 s	3.17 s	0.19 s	9.69 s	0.27 s
9	1.06 s	0.09 s	3.37 s	0.12 s	> 3600 s	0.27 s
10	0.92 s	0.15 s	203 s	0.19 s	> 3600 s	0.27 s

TABLEAU 4.11 – Temps de calcul moyen de IMPMCG et GRASP+ILP sur des petites instances

Afin d'évaluer le temps de calcul pour résoudre IMPMCG, 30 instances ont été générées

avec $m \in \{20, 30, 40\}$ et $n = 4$ cibles, de la même manière que les 100 instances précédentes. Le tableau 4.11 compare les temps de calcul de la résolution de IMPMCG et la résolution avec GRASP+ILP (abrégé G+ILP) sur ces instances. Une limite de temps de calcul a été fixée à 3600 secondes. Le temps de calcul est significativement plus élevé avec l'approche IMPMCG, et le nombre d'instances qui requièrent plus d'une heure de calcul augmente rapidement avec le nombre de capteurs. Or, toutes ces instances ont été résolues en moins d'une seconde avec l'approche GRASP+ILP. Comme dans (Rossi *et al.* 2012), ceci montre clairement l'efficacité des approches à base de génération de colonnes.

4.6 Conclusion

Dans le cadre de missions multiples, il est important de préserver les capacités des capteurs d'une zone d'intérêt à surveiller ultérieurement. Dans ce chapitre, nous avons étudié un problème décomposé en trois sous-problèmes. Le rôle du premier, MRC, est de vérifier l'existence d'un ordonnancement réalisable d'activités de capteurs. Le second sous-problème, MCG, consiste à maximiser la garantie de couverture, la durée pour laquelle le réseau est capable de surveiller tout point de la zone d'intérêt à l'issue de la mission courante. Enfin, le troisième, MEC, consiste à minimiser l'énergie totale consommée, tout en préservant la garantie de couverture maximale. La méthode présentée se déroule en deux grandes phases : la discrétisation, et la résolution d'un problème d'ordonnancement d'activités des capteurs. La discrétisation reformule le problème général afin de s'affranchir des données géométriques et de permettre l'utilisation d'outils de résolution. Elle produit ainsi l'instance d'un problème d'ordonnancement d'activités de capteurs, dont la taille peut être réduite par la suppression de faces à couvrir et des fusions de fenêtres de temps, sans modifier le problème. La résolution du problème d'ordonnancement repose sur la génération de colonnes et est décomposée en trois étapes. La première étape consiste à générer un ensemble de couvertures initial, par la résolution de MRC, afin de vérifier l'existence d'un ordonnancement réalisable. Les deuxième et troisième étapes consistent à résoudre MCG et MEC successivement. Les problèmes maîtres de MRC, MCG et MEC ont été formulés de manière à partager le même ensemble de couvertures et le même problème auxiliaire. Le problème auxiliaire est lui-même décomposable en fonction des fenêtres de temps. Chaque sous-problème auxiliaire, formulé comme la variante d'un problème de couverture d'ensembles, peut être résolu avec GRASP, et le cas échéant, avec un solveur de programmes linéaires en variables binaires.

Des instances jusqu'à 150 capteurs et 15 cibles sont résolues en moins de 20 secondes en moyenne, lorsque les cibles ne sont pas sujettes à incertitude. Dans le cas où les cibles sont sujettes à incertitude, ces mêmes instances sont résolues en moins de 30 secondes en moyenne. La procédure de réduction d'instances permet de supprimer de nombreuses fenêtres de temps lorsque le nombre de cibles est élevé ou lorsqu'elles sont sujettes à incertitude. Il s'ensuit que le nombre de problèmes auxiliaires est diminué jusqu'à 60% en moyenne pour le cas où les cibles sont sujettes à incertitude.

Les perspectives offertes par ces travaux incluent la prise en compte des coûts de communication et de l'acheminement des données de surveillance vers une station de base. De nouveaux critères pourraient être pris en compte, comme une mesure de robustesse à l'instar du problème présenté au chapitre 2.

Chapitre 5

Extensions, perspectives et conclusion

Contenu

Les précédents chapitres présentent des problèmes d'ordonnancement d'activités de capteurs pouvant faire l'objet de nombreuses extensions, ce qui ouvre des perspectives pour la résolution de problèmes tenant compte de multiples aspects de la réalité. Le présent chapitre propose trois extensions pertinentes au problème MSR du chapitre 3.

Les capteurs, équipés d'un module de communication, peuvent acheminer les données de surveillance vers une station de base, afin que l'exploitant puisse analyser et traiter ces données. La communication énergétique liée aux communications est prise en compte dans l'extension présentée dans la section 5.1. Dans la section 5.2, l'extension présentée permet de construire un ordonnancement robuste pour la surveillance de plusieurs cibles. Enfin, l'extension proposée dans la section 5.3 prend en compte à la fois l'incertitude spatiale et temporelle pour le suivi d'une cible.

5.1 Coûts de communication

Dans le problème présenté au chapitre 3, noté MSR (Maximize Stability Radius), les capteurs ont pour mission de surveiller une cible, pour laquelle la trajectoire spatiale est connue avec certitude, mais dont les temps de passage en chacun de ses points sont incertains. Le but est de construire un ordonnancement robuste d'activités de capteurs, garantissant la surveillance de la cible, malgré une avance ou un retard par rapport aux temps de passage prévisionnels.

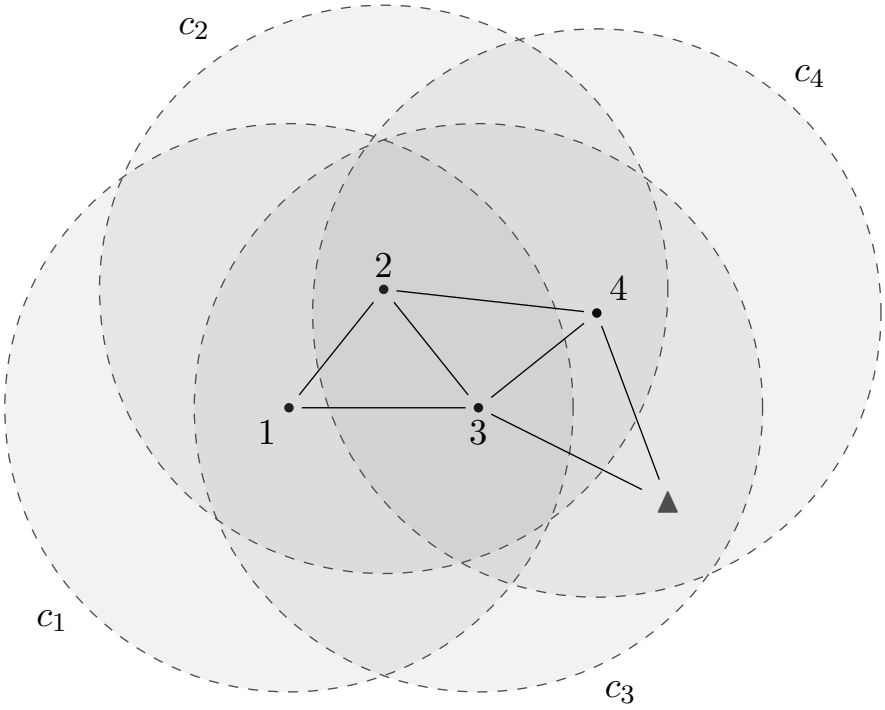


FIGURE 5.1 – Exemple de graphe de communication, où ▲ est une station de base

Dans cette section, nous présentons une extension de ce problème dans laquelle les coûts d'acheminement des données vers une station de base sont pris en compte. La station de base est typiquement un relais vers un centre de contrôle dont le but est de collecter et traiter les données envoyées par les capteurs. Les capteurs disposent d'un module de communication sans fil capable d'émettre et recevoir des données (wifi, bluetooth, etc.). Les capacités des batteries des capteurs sont désormais exprimées en unités d'énergie. Lors d'une activité de surveillance, un capteur collecte β unités de données par unité de

temps, et consomme sur sa batterie e^S unités d'énergie par unité de temps (il s'agit donc d'une puissance). Un capteur i est capable de transmettre des données aux capteurs se situant dans sa zone de communication c_i. Sans perte de généralité, nous considérons que la zone de communication d'un capteur est un disque de rayon R^C centré sur le capteur. La transmission d'une unité de données coûte e^T unités d'énergie, et la réception e^R. La station de base est supposée positionnée à l'intérieur d'au moins une zone de communication. Un récapitulatif des données d'entrée est présenté dans le tableau 5.1.

I	Ensemble des capteurs $\{1, \ldots, m\}$
s_i	Zone de couverture du capteur i
c_i	Zone de communication du capteur i
E_i	Capacité du capteur i en unités d'énergie
$\mathcal{T}(t)$	Position de la cible à l'instant t
e^S	Energie consommée par un capteur pour la surveillance d'une cible
e^T	Energie consommée par un capteur pour l'envoi de données
e^R	Energie consommée par un capteur pour la réception de données
β	Quantité de données de surveillance par unité de temps

TABLEAU 5.1 – Données d'entrée de MSRC

Pour cette extension, qu'on notera MSRC (Maximize Stability Radius with Communication), la méthode proposée est basée sur celle du chapitre 3. Elle se déroule en deux phases : la discrétisation, et la résolution d'un problème d'ordonnancement d'activités de capteurs. La discrétisation, qui consiste à reformuler les données du problème, est décrite en détail dans le chapitre 2 et la section 3.2.1 (page 35).

K	Ensemble des fenêtres de temps $\{1, \ldots, p\}$
$S_j(k) \subseteq I$	Ensemble des capteurs candidats couvrant la cible pendant la fenêtre de temps $k \in K \cup \{0, p+1\}$
t_k	Tick d'indice $k \in K \cup \{p+1\}$
σ_k	Valeur déterminant la catégorie du tick k (entrant ou sortant)
Δ_k	Durée de la fenêtre de temps $k \in K$
$N(i)$	Ensemble des nœuds (capteurs, station de base) capables de communiquer avec le capteur i

TABLEAU 5.2 – Données résultant de la discrétisation

L'objectif de MSRC est identique à celui de MSR, à savoir maximiser le rayon de stabilité, noté ρ. Le rayon de stabilité d'un ordonnancement $\mathcal{S}$ est défini par la plus grande

variation des ticks t_k qui ne compromet pas la capacité de $\mathcal{S}$ à couvrir continuellement la cible sans la perdre de vue. Autrement dit, l'objectif est de construire un ordonnancement qui résiste le mieux possible aux avances ou retards éventuels de la cible par rapport aux temps de passage prévisionnels.

Afin de résoudre MSRC, on ajoute à la discrétisation la construction d'un graphe de communication, afin de déterminer entre quels capteurs peuvent transiter les flux de données. L'ensemble des sommets est constitué des capteurs et de la station de base. Les arêtes représentent les liens de communication. La discrétisation produit alors les ensembles $N(i)$, ensemble des nœuds (c'est-à-dire des capteurs et/ou la station de base) capables de communiquer avec le capteur i. Un capteur j appartient à $N(i)$ si la distance entre les capteurs i et j est inférieure au rayon de communication R^C. Ainsi, une arête est créée entre deux nœuds distincts i et j si et seulement si $j \in N(i)$. Les données résultant de la discrétisation sont résumées dans le tableau 5.2.

À l'issue de la discrétisation, la résolution de MSRC se déroule suivant le même schéma que la méthode proposée pour MSR. Celle-ci consiste en une dichotomie sur le rayon de stabilité ρ. Premièrement, une borne supérieure UB sur ρ est calculée afin de réduire l'espace de recherche de la dichotomie à l'intervalle $[0, \text{UB}]$. L'espace de recherche est alors représenté par une liste ordonnée $\mathcal{D}$ contenant toutes les distances positives $t_{k'} - t_k < UB$ où t_k est un tick entrant et $t_{k'}$ un tick sortant. Puis, à chaque itération de la dichotomie, un problème de décision est résolu. Le problème de décision consiste, pour une valeur de ρ donnée, à déterminer s'il existe un ordonnancement réalisable dont le rayon de stabilité est d'au moins ρ.

Pour rappel (section 3.3.1, page 44), dans le cas du problème MSR, le problème de décision peut être formulé comme un problème de transport. Pour $\rho = 0$, une instance $\mathcal{I}_0$ du problème de transport est construite en attribuant aux capteurs le rôle de fournisseurs (de capacité E_i) et aux fenêtres de temps le rôle de clients (demande Δ_k). L'instance est équilibrée en créant un client fictif dont la demande est $\Delta_{p+1} = \sum_{i \in I} E_i - \sum_{k \in K} \Delta_k$. Les coûts de transport c_{ik} ont le rôle de pénalités afin d'empêcher l'activation de capteurs non-candidats. Ainsi $c_{ik} = 1$ si le capteur i n'est pas candidat pour la fenêtre de temps k, 0 sinon. Si la valeur optimale de la fonction objectif est nulle, alors il existe un ordonnancement réalisable pour $\mathcal{I}_0$. Pour $\rho > 0$, une instance $\mathcal{I}_\rho$ est une instance dans laquelle les ticks de $\mathcal{I}_0$ ont été déplacés de telle sorte que s'il existe un ordonnancement réalisable pour $\mathcal{I}\rho$, alors cet ordonnancement a un rayon de stabilité d'au moins ρ pour $\mathcal{I}_0$. L'instance $\mathcal{I}_\rho$ est construite de la manière suivante. On retarde les ticks entrants de $\mathcal{I}_0$ de ρ unités de temps,

tandis que les ticks sortants sont avancés de ρ unités de temps. Si un tick entrant et un tick sortant se rencontrent, alors leur position est échangée dans la séquence des ticks, et l'ensemble des capteurs candidats $S(k)$ est mis à jour :

$$S(k) \leftarrow S(k-1) \cap S(k+1)$$

Enfin, comme pour l'instance $\mathcal{I}_0$, les capteurs jouent le rôle de fournisseurs et les fenêtres de temps prennent le rôle de clients (avec les nouvelles valeurs de Δ_k). Les coûts c_{ik} de transport sont attribués de la même manière.

Le problème de décision de MSRC ne peut pas être formulé comme un problème de transport, puisqu'il doit tenir compte des coûts de communication. Toutefois, il peut être formulé comme un programme linéaire en variables continues. De manière similaire à MSR, une instance $\mathcal{I}_\rho$ est une instance dans laquelle les ticks de $\mathcal{I}_0$ ont été déplacés de telle manière que s'il existe un ordonnancement réalisable pour $\mathcal{I}_\rho$, alors cet ordonnancement a un rayon de stabilité d'au moins ρ pour l'instance $\mathcal{I}_0$. Les ticks entrants sont retardés de ρ unités de temps et les ticks sortants sont avancés de ρ unités de temps. Si deux ticks entrants et sortants se rencontrent à l'issue de ces déplacements, alors ils sont échangés. On obtient donc de nouvelles valeurs des durées de fenêtres de temps Δ_k pour $\mathcal{I}_\rho$.

Le programme linéaire $\mathrm{LP}_{\mathrm{MSRC}}$ contient les variables suivantes :

— x_{ik} : durée de surveillance attribuée au capteur i pendant la fenêtre de temps k

— $f_{ii'}$: quantité de données circulant du capteur i au capteur (ou station de base) i'

— δ : augmentation de la valeur de ρ

On définit la constante σ_k :

$$\sigma_k = \begin{cases} 1 & \text{si } t_k \text{ est un tick entrant} \\ -1 & \text{si } t_k \text{ est un tick sortant} \end{cases}$$

L'objectif est de maximiser l'augmentation de la valeur de ρ. La contrainte (5.2) impose que l'énergie totale consommée par les capteurs n'excède pas leur capacité. La contrainte (5.3) assure la conservation des flots de données. La contrainte (5.4) garantit que la demande de temps de couverture de la cible est satisfaite.

Pour résoudre le problème de décision de MSRC, une solution réalisable pour $\mathrm{LP}_{\mathrm{MSRC}}$ est suffisante. Il n'est donc pas nécessaire d'obtenir la valeur optimale de δ, sauf à la dernière itération de la dichotomie, où l'on souhaite calculer la valeur optimale de ρ. Tout comme MSR, le problème MSRC peut être résolu en un nombre pseudo-polynomial d'itérations.

Modèle $\mathrm{LP}_{\mathrm{MSRC}}$

$$\max \delta \tag{5.1}$$

$$e^S \sum_{k\in K|i\in S(k)} x_{ik} + e^T \sum_{i'\in N(i)} f_{ii'} + e^R \sum_{i'\in N^-(i)} f_{i'i} \leq E_i \qquad \forall i \in I \tag{5.2}$$

$$\sum_{k\in K|i\in S(k)} x_{ik} + \sum_{i'\in N(i)} f_{i'i} = \sum_{i'\in N(i)} f_{ii'} \qquad \forall i \in I \tag{5.3}$$

$$\sum_{i\in S(k)} x_{ik} = \Delta_k + (\sigma_{k+1} - \sigma_k)\delta, \forall k \in K \tag{5.4}$$

$$\delta \geq 0 \tag{5.5}$$

$$x_{ik} \geq 0 \qquad \forall k \in K, i \in S(k) \tag{5.6}$$

$$f_{ii'} \geq 0 \qquad \forall i \in I, i' \in N(i) \tag{5.7}$$

En effet, le problème de décision peut être résolu à l'aide d'un algorithme polynomial tel qu'un algorithme de point intérieur pour les programmes linéaires en variables continues. La dichotomie effectue un nombre logarithmique d'itérations en fonction du nombre de fenêtres de temps, soit $\mathcal{O}(\log |K|)$. Or, il est possible de concevoir une instance dont le nombre de fenêtres de temps est aussi grand que l'on veut, sans augmenter le nombre de capteurs ou de cibles (section 3.3.2, page 50). La méthode globale s'exécute donc en temps pseudo-polynomial.

Bornes supérieures

Les bornes supérieures pour MSR sont valides pour MSRC. En effet, MSR peut être vue comme une relaxation de MSRC, dans laquelle les contraintes de communication sont ignorées, et les constantes prennent les valeurs $e^S = 1$, $e^T = 0$ et $e^R = 0$. Ainsi, les bornes UB1, UB1′, UB2 et UB2′ présentées dans la section 3.2.3 (page 38) s'appliquent à MSRC.

Implémentation

Bien que l'implémentation n'ait pas été réalisée, nous disposons de tous les outils nécessaires pour l'effectuer. En effet, l'implémentation d'un solveur pour MSRC peut se baser sur le solveur de MSR. Au lieu de faire appel à un solveur de problème de transport, la dichotomie appelle un solveur de programme linéaire en variables continues tel

que CPLEX, afin de résoudre $\mathrm{LP}_{\mathrm{MSRC}}$. La solution obtenue peut être codée comme un ordonnancement d'arbres, où les feuilles sont les capteurs actifs, les nœuds intermédiaires sont des capteurs relais et la racine est la station de base. Cet ordonnancement peut être construit à l'aide d'un algorithme comme celui proposé dans (Liu *et al.* 2011). Les variables $f_{ii'}$ du programme linéaire $\mathrm{LP}_{\mathrm{MSRC}}$ représentent un budget de communication. Un capteur i envoie ses données à n'importe quel capteur i' tel que $f_{ii'} > 0$, jusqu'à saturation, c'est-à-dire jusqu'à ce que la quantité de données envoyées atteigne $f_{ii'}$. Lorsqu'un lien de communication est saturé, les données peuvent en emprunter un autre non-saturé, jusqu'à ce que tous les liens soient saturés. La contrainte (5.3) assure qu'il existe toujours un lien de communication non-saturé.

5.2 Surveillance multi-cibles

Dans le chapitre 3, nous présentons le problème MSR, consistant à planifier un ordonnancement robuste d'activités de capteurs de manière à surveiller une seule cible. Dans la présente section, nous étudions une extension appelée MSRMT (Maximize Stability Radius with Multiple Targets) dans laquelle le réseau de capteurs doit surveiller plusieurs cibles. L'ensemble des cibles est noté J. La trajectoire spatiale de chaque cible est connue à l'avance et modélisée par une fonction vectorielle continue $\mathcal{T}_j : \mathbb{R} \to \mathbb{R}^2$, définie par $\mathcal{T}_j(t)$ où t est une variable de temps. Autrement dit, à un instant t, la cible est attendue à la position $\mathcal{T}_j(t)$. Toutefois, comme dans le cas de MSR, les temps de passage sont incertains.

Les données du problème sont reformulées grâce à la *discrétisation* présentée au chapitre 2. Pour cette extension, nous proposons l'utilisation de la variante de la discrétisation suivant les cibles (section 2.5, page 27). Ainsi, les ensembles de capteurs candidats sont référencés par cible. Les données résultant de la discrétisation sont résumées dans le tableau 5.3.

Tout comme pour MSR, l'objectif de MSRMT est de construire un ordonnancement maximisant le rayon de stabilité ρ. Soit $\mathcal{S}$ un ordonnancement, son rayon de stabilité est la plus grande variation des ticks t_k qui ne compromet pas la capacité de $\mathcal{S}$ à garantir la couverture de la cible.

Le problème de décision de MSRMT consiste à déterminer s'il existe un ordonnancement réalisable tel que le rayon de stabilité est au moins ρ. Ce problème de décision est modélisé par MP, un programme linéaire en variables continues, formulé comme le problème maître d'un algorithme de génération de colonnes. Une *couverture* est un sous-ensemble des

I	Ensemble des capteurs $\{1, \ldots, m\}$
J	Ensemble des cibles $\{1, \ldots, n\}$
E_i	Durée de vie de la batterie du capteur $i \in I$
K	Ensemble des fenêtres de temps $\{1, \ldots, p\}$
t_k	Date de début de la fenêtre de temps $k \in K$
Δ_k	Durée de la fenêtre de temps $k \in K$
σ_k	1 si le tick k est entrant, -1 s'il est sortant
$S_j(k)$	Ensemble des capteurs candidats à la couverture de la cible j pendant la fenêtre de temps k, c-à-d entre les ticks t_k et t_{k+1}, $k \in K$

TABLEAU 5.3 – Données d'entrée du problème d'ordonnancement

capteurs couvrant toutes les cibles. Ainsi, une couverture est valide pour k si pour chaque cible j, au moins un capteur de $S_j(k)$ est activé. Soit $\mathcal{C}$ l'ensemble de toutes les couvertures et $\mathcal{C}_k$ l'ensemble des couvertures valides pour la fenêtre de temps k. On définit la constante $a_{ic} = 1$ si le capteur i appartient à la couverture c, 0 sinon. Le programme linéaire MP consiste à déterminer les durées d'activation des couvertures.

— d_c^k durée de surveillance attribuée à la couverture c pendant la fenêtre de temps k

— δ augmentation de la valeur de ρ

Modèle MP

$$\max \delta \tag{5.8}$$

$$\sum_{k \in K} \sum_{c \in \mathcal{C}(k)} a_{ic} d_c^k \leq E_i \qquad \forall i \in I[\pi_i] \tag{5.9}$$

$$\sum_{c \in \mathcal{C}(k)} d_c^k = \Delta_k + (\sigma_{k+1} - \sigma_k)\delta \qquad \forall k \in K[\mu_k] \tag{5.10}$$

$$\delta \geq 0 \tag{5.11}$$

$$d_c^k \geq 0 \qquad \forall k \in K, c \in \mathcal{C}(k) \tag{5.12}$$

L'objectif est de maximiser l'augmentation du rayon de stabilité ρ. Toutefois, pour obtenir un ordonnancement réalisable, il n'est pas nécessaire de résoudre MP à l'optimalité puisqu'obtenir une solution réalisable pour MP suffit à résoudre le problème de décision de MSRMT. La contrainte (5.9) assure que la durée totale d'activité des capteurs ne dépasse pas leur capacité. La contrainte (5.10) impose la couverture des cibles à chaque instant de toutes les fenêtres de temps.

Le problème auxiliaire consiste à construire des couvertures de coût réduit positif. Il peut être décomposé en $|K|$ sous-problèmes indépendants, notés PP_k.

Modèle PP_k

$$\max \; -\sum_{i \in I} \pi_i a_{ic} - \mu_k \tag{5.13}$$

$$\sum_{i \in S_j(k)} a_{ic} \geq 1 \qquad \forall j \in J \tag{5.14}$$

$$a_{ic} \in \{0,1\} \qquad \forall i \in I \tag{5.15}$$

L'objectif est de maximiser le coût réduit de la nouvelle couverture. La contrainte (5.14) garantit la couverture de toutes les cibles.

La formulation du problème de décision est conçue pour l'exécution d'un algorithme de génération de colonnes. Afin de trouver le rayon de stabilité optimal, une méthode appropriée est la dichotomie. À chaque itération de la dichotomie, on résout le problème de décision à l'aide d'un algorithme de génération de colonnes.

A la rencontre de deux ticks t_k et t_{k+1}, une opération sur les ensembles de capteurs candidats est effectuée.

$$S_j(k) \leftarrow S_j(k-1) \cap S_j(k+1)$$

Les couvertures existantes d'une fenêtre de temps peuvent être gardées lorsque l'ensemble de capteurs candidats ne change pas. En revanche, il est peut-être nécessaire de retirer les colonnes d_k^c car elles pourraient ne plus être valides avec les nouveaux $S_j(k)$.

Implémentation

L'implémentation d'une méthode pour MSRMT peut être basée sur l'implémentation du solveur de MSR. La résolution d'un problème de transport est remplacée par un algorithme de génération de colonnes, qui calcule une solution réalisable de MP pour répondre au problème de décision. À l'issue de la dichotomie le programme linéaire MP est résolu à l'optimalité pour trouver le rayon de stabilité.

5.3 Incertitude spatiale et temporelle

Les problèmes présentés dans les chapitres 3 et 4 introduisent respectivement l'incertitude temporelle et spatiale. Dans cette section, nous présentons une extension de MSR (chapitre 3) qui prend également en compte l'incertitude spatiale. Cette extension est notée MSRSU (Maximize Stability Radius under Spatial Uncertainty). Le réseau de capteurs doit surveiller une cible dont la trajectoire est modélisée par une fonction vectorielle continue $\mathcal{T} : \mathbb{R} \to \mathbb{R}^2$, définie par $\mathcal{T}(t)$ où t est une variable de temps. La position de la cible est sujette à incertitude : à chaque instant t, elle peut se situer à une distance δ de sa position attendue $\mathcal{T}(t)$. On considère alors un disque d'incertitude de rayon δ à couvrir à chaque instant, où δ est un paramètre d'entrée du problème. Les temps de passage de la cible sont également sujets à incertitude : la cible (ainsi que son disque d'incertitude) peut arriver en avance ou en retard d'une durée ρ par rapport aux dates prévisionnelles. Le but est de construire un ordonnancement à énergie minimale, capable de couvrir la cible malgré cette avance ou ce retard, et malgré l'incertitude spatiale. Afin que le réseau de capteurs consomme le moins d'énergie possible, un seul capteur peut être activé à chaque instant.

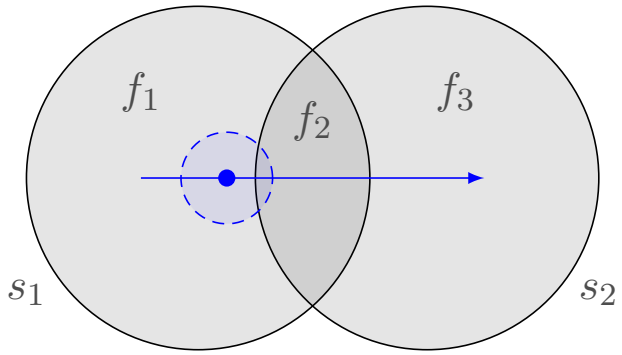


FIGURE 5.2 – Exemple d'instance simple de MSRSU

À l'instar des précédents problèmes étudiés, celui-ci est reformulé grâce à la discrétisation (chapitre 2). Rappelons que la zone de surveillance peut être partitionnée en surfaces disjointes appelées *faces*. Dans le cas où les positions ne sont pas sujettes à incertitude, la trajectoire de la cible peut être formulée comme une séquence des faces traversées. Dans le cas contraire, la trajectoire est une séquence d'ensembles de faces traversées par le disque d'incertitude. Chaque fois que le disque d'incertitude franchit la frontière d'une face, la discrétisation crée un *tick*, date de ce franchissement noté t_k où k est son indice. L'intervalle de temps entre deux ticks t_k et t_{k+1} consécutifs est une *fenêtre de*

temps, d'une durée notée Δ_k. Le temps est ainsi partitionné en fenêtres de temps disjointes, durant lesquelles l'ensemble des faces à couvrir est le même. En résumé, à chaque fenêtre de temps k et chaque cible j on associe un ensemble d'ensembles de capteurs candidats $\Sigma(k) \subseteq \{S|S \subseteq I\}$.

Un exemple est illustré sur la figure 5.2. Dans cette instance, le temps peut être partitionné en 5 fenêtres de temps et 6 ticks. Dans la première fenêtre (entre les ticks t_1 et t_2), le disque d'incertitude se situe intégralement dans la face f_1, couverte par le capteur 1. Lorsque le disque franchit la frontière de s_2 (la zone de couverture du capteur 2), la seconde fenêtre débute (au tick t_2) et le disque chevauche à la fois les faces f_1 et f_2. Dans les fenêtres suivantes, le disque chevauche $\{f_2\}$, puis $\{f_2, f_3\}$, et enfin $\{f_3\}$. À l'issue de la discrétisation, nous admettrons que les données produites sont les suivantes :

$$\begin{array}{lll} \Sigma(1) = \{\{s_1\}\} & \Delta_1 = 1 & \sigma_1 = -1 \\ \Sigma(2) = \{\{s_1\}, \{s_1, s_2\}\} & \Delta_2 = 1 & \sigma_2 = 1 \\ \Sigma(3) = \{\{s_1, s_2\}\} & \Delta_3 = 1 & \sigma_3 = 1 \\ \Sigma(4) = \{\{s_1, s_2\}, \{s_2\}\} & \Delta_4 = 1 & \sigma_4 = -1 \\ \Sigma(5) = \{\{s_2\}\} & \Delta_5 = 1 & \sigma_5 = -1 \\ & & \sigma_6 = 1 \end{array}$$

Montrons que cette instance de MSRSU peut être formulée comme une instance de MSR. Les contraintes de MSRSU imposent qu'un seul capteur ne peut être activé à chaque instant. Par conséquent, si le réseau de capteurs doit surveiller deux faces, il doit nécessairement activer un capteur qui couvre ces deux faces. Les capteurs qui ne peuvent couvrir toutes les faces requises dans une fenêtre de temps k peuvent être supprimés des ensembles $\Sigma(k)$. Ainsi, l'ensemble des capteurs candidats dans la fenêtre de temps k est l'intersection des ensembles de $\Sigma(k)$, soit $S(k) = \cap\Sigma(k)$. L'instance de MSRSU est alors reformulée en une instance de MSR :

$$
\begin{array}{lll}
S(1) = \{s_1\} & \Delta_1 = 1 & \sigma_1 = -1 \\
S(2) = \{s_1\} & \Delta_2 = 1 & \sigma_2 = 1 \\
S(3) = \{s_1, s_2\} & \Delta_3 = 1 & \sigma_3 = 1 \\
S(4) = \{s_2\} & \Delta_4 = 1 & \sigma_4 = -1 \\
S(5) = \{s_2\} & \Delta_5 = 1 & \sigma_5 = -1 \\
 & & \sigma_6 = 1
\end{array}
$$

Remarquons que s'il existe deux faces qui ne sont couvertes par aucun capteur en commun, alors le problème MSRSU est non-réalisable. Cette situation se rencontre notamment lorsque le rayon du disque d'incertitude est suffisamment grand pour chevaucher des faces très éloignées.

Implémentation

Puisqu'une instance de MSRSU peut être transformée en une instance de MSR, le solveur de MSR peut être réutilisé directement. Toutefois, la discrétisation doit inclure la procédure de transformation décrite dans la présente section.

5.4 Conclusion

Les réseaux de capteurs sans fil ont suscité une attention croissante ces dernières années, et constituent désormais un domaine à part entière dans la littérature scientifique. Ces appareils, typiquement équipés d'un dispositif de mesure, d'un module de communication et d'une batterie, offrent beaucoup d'applications, comme la suivi de véhicules ou la surveillance d'un champ de bataille. Dans le chapitre 1, nous avons apporté une vue d'ensemble des techniques d'optimisation dans les réseaux de capteurs sans fil. Beaucoup de travaux ont été réalisés pour optimiser l'usage des capteurs pour le suivi de cibles fixes à l'aide d'outils de la recherche opérationnelle. Toutefois, les techniques d'optimisation sont moins nombreuses pour le suivi de cibles mobiles. Cette thèse s'inscrit ainsi dans la suite logique de travaux existants, en répondant aux perspectives offertes par les travaux précédents.

Avant le début d'une mission de surveillance, les données dont nous disposons sont typiquement les positions des capteurs et des prévisions sur la trajectoire de cibles à surveiller. Dans le chapitre 2, nous avons présenté une reformulation de ces données afin de les rendre

exploitables par des méthodes d'optimisation. Elle constitue un préalable indispensable aux méthodes présentées dans cette thèse, et se décline en deux variantes pour convenir à tous les problèmes. L'une des variantes est complétée par une procédure de réduction de la taille d'une instance, qui permet de réduire également le temps de calcul total dans la plupart des cas.

Le problème présenté dans le chapitre 3 noté MSR consiste à planifier un ordonnancement robuste d'activités de capteurs à énergie minimale afin de surveiller une cible dont la trajectoire spatiale est connue mais dont les temps de passage sont incertains. Ceci correspond notamment au cas où la cible à surveiller est un train sur une voie ferrée ou un véhicule terrestre se déplaçant sur une route. Nous avons introduit une mesure de robustesse, le *rayon de stabilité*, objectif de notre problème à maximiser. La méthode proposée est une dichotomie sur le rayon de stabilité, s'exécutant en un nombre pseudo-polynomial d'itérations dans le cas général, et en temps polynomial lorsque la trajectoire de la cible est une ligne polygonale. Toutefois, dans de nombreux cas, nous avons constaté qu'il était possible de s'affranchir de la dichotomie grâce au calcul de bornes supérieures rencontrant souvent la valeur optimale du rayon de stabilité. Les temps de calcul sont ainsi très raisonnables puisque des instances comportant jusqu'à 1000 capteurs sont résolues en moins de 15 secondes en moyenne.

Le chapitre 4 introduit un problème où le réseau de capteurs doit surveiller plusieurs cibles et préserver sa capacité à surveiller une zone d'intérêt à l'issue de la mission courante. Le but est de construire un ordonnancement d'activités de capteurs qui maximise cette capacité de surveillance, et qui, dans le même temps, minimise la consommation totale d'énergie. De plus, la position des cibles est sujette à incertitudes, c'est-à-dire qu'elle peut s'écarter d'une certaine distance par rapport à la position attendue. Ce problème, prouvé NP-difficile, est décomposé en trois sous-problèmes successifs, formulés comme le problème maître d'un algorithme de génération de colonnes. Les trois problèmes maîtres partagent le même problème auxiliaire, formulé comme une variante du problème de couverture d'ensembles. Cette décomposition permet en particulier d'exploiter les colonnes d'un sous-problème pour un autre sous-problème. La méthode proposée est une matheuristique, basée sur la génération de colonnes et accélérée à l'aide d'une métaheuristique GRASP.

Enfin, nous avons présenté dans le chapitre 5, trois exemples d'extensions au problème MSR du chapitre 3. La première extension prend en compte l'acheminement des données de surveillance vers une station de base, afin que l'exploitant puisse les traiter. Dans la

deuxième extension, le réseau de capteurs est capable de surveiller plusieurs cibles, sous les mêmes contraintes que MSR. Enfin, la troisième extension considère l'incertitude spatiale, en plus de l'incertitude temporelle, pour la surveillance d'une cible. Pour chacune des extensions, nous avons esquissé une méthode afin d'ouvrir des perspectives de recherche.

Je remercie la Direction Générale de l'Armement (DGA) de m'avoir accordé un financement pour ces travaux de recherche.

Bibliographie

"10 Emerging Technologies That Will Change the World" (2003). In : *MIT Technology Review*. URL : http://www2.technologyreview.com/news/401775/10-emerging-technologies-that-will-change-the/2/.

Alfieri, A. *et al.* (2007). "Maximizing system lifetime in wireless sensor networks". In : *European Journal of Operational Research* 181.1, p. 390–402. DOI : 10.1016/j.ejor.2006.05.037.

Bari, Ataul *et al.* (2012). "Design of fault tolerant wireless sensor networks satisfying survivability and lifetime requirements". In : *Computer Communications* 35.3, p. 320–333. DOI : 10.1016/j.comcom.2011.10.006.

Berman, P. *et al.* (2004). "Power efficient monitoring management in sensor networks". In : *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/wcnc.2004.1311452.

Bertsimas, Dimitris, David B. Brown et Constantine Caramanis (2011). "Theory and Applications of Robust Optimization". In : *SIAM Rev.* 53.3, p. 464–501. DOI : 10.1137/080734510.

Billaut, J.C., A. Moukrim et E. Sanlaville, éds. (2005). *Flexibilité et robustesse en ordonnancement*. Hermès.

Cardei, M. *et al.* (2005a). "Energy-efficient target coverage in wireless sensor networks". In : *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/infcom.2005.1498475.

Cardei, M. *et al.* (2005b). "Maximum network lifetime in wireless sensor networks with adjustable sensing ranges". In : *WiMob'2005), IEEE International Conference on Wi-*

reless And Mobile Computing, Networking And Communications, 2005. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/wimob.2005.1512935.

CARDEI, Mihaela et Ding-Zhu DU (2005c). "Improving Wireless Sensor Network Lifetime through Power Aware Organization". In : *Wireless Netw* 11.3, p. 333–340. DOI : 10.1007/s11276-005-6615-6.

CARRABS, Francesco *et al.* (2015a). "A hybrid exact approach for maximizing lifetime in sensor networks with complete and partial coverage constraints". In : *Journal of Network and Computer Applications* 58, p. 12–22. DOI : 10.1016/j.jnca.2015.08.018.

CARRABS, Francesco *et al.* (2015b). "Maximizing lifetime in wireless sensor networks with multiple sensor families". In : *Computers & Operations Research* 60, p. 121–137. DOI : 10.1016/j.cor.2015.02.013.

CASTAÑO, F. *et al.* (2013). "On the use of multiple sinks to extend the lifetime in connected wireless sensor networks". In : *Electronic Notes in Discrete Mathematics* 41, p. 77–84. DOI : 10.1016/j.endm.2013.05.078.

CASTAÑO, Fabian, André ROSSI et Marc SEVAUX (2013). "A hybrid Column Generation-GRASP heuristic to extend the lifetime in wireless sensor networks with connectivity and coverage constraints". In : *ROADEF, 14ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision.* Troyes, France. URL : https://hal.archives-ouvertes.fr/hal-00789529.

CASTAÑO, Fabian *et al.* (2014). "A column generation approach to extend lifetime in wireless sensor networks with coverage and connectivity constraints". In : *Computers & Operations Research* 52, p. 220–230. DOI : 10.1016/j.cor.2013.11.001.

CASTAÑO, Fabian *et al.* (2015). "Exact approaches for lifetime maximization in connectivity constrained wireless multi-role sensor networks". In : *European Journal of Operational Research* 241.1, p. 28–38. DOI : 10.1016/j.ejor.2014.08.013.

CERULLI, R., R. De DONATO et A. RAICONI (2012). "Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges". In : *European Journal of Operational Research* 220.1, p. 58–66. DOI : 10.1016/j.ejor.2012.01.046.

CHENG, M.X., Lu RUAN et Weili WU (2005). "Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks". In : *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/infcom.2005.1498547.

CHENG, Weifang *et al.* (2007). "Maximal Coverage Scheduling in Randomly Deployed Directional Sensor Networks". In : *2007 International Conference on Parallel Processing Workshops (ICPPW 2007)*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/icppw.2007.51.

DANTZIG, George B. et Philip WOLFE (1960). "Decomposition Principle for Linear Programs". In : *Operations Research* 8.1, p. 101–111. DOI : 10.1287/opre.8.1.101.

DELICATO, Flávia *et al.* (2006). "An efficient heuristic for selecting active nodes in wireless sensor networks". In : *Computer Networks* 50.18, p. 3701–3720. DOI : 10.1016/j.comnet.2006.04.005.

DEMIGHA, Oualid, Walid-Khaled HIDOUCI et Toufik AHMED (2013). "On Energy Efficiency in Collaborative Target Tracking in Wireless Sensor Network: A Review". In : *IEEE Communications Surveys & Tutorials* 15.3, p. 1210–1222. DOI : 10.1109/surv.2012.042512.00030.

DENG, Xiu *et al.* (2012). "Transforming Area Coverage to Target Coverage to Maintain Coverage and Connectivity for Wireless Sensor Networks". In : *International Journal of Distributed Sensor Networks* 2012, p. 1–12. DOI : 10.1155/2012/254318.

DESROSIERS, Jacques et Marco E. LÜBBECKE (2005). "A Primer in Column Generation". In : *Column Generation*. Springer Science + Business Media, p. 1–32. DOI : 10.1007/0-387-25486-2_1.

DEZSŐ, Balázs, Alpár JÜTTNER et Péter KOVÁCS (2011). "LEMON – an Open Source C++ Graph Template Library". In : *Electronic Notes in Theoretical Computer Science* 264.5, p. 23–45. DOI : 10.1016/j.entcs.2011.06.003.

FEO, Thomas A. et Mauricio G. C. RESENDE (1995). "Greedy Randomized Adaptive Search Procedures". In : *Journal of Global Optimization* 6.2, p. 109–133. DOI : 10.1007/bf01096763.

FORD, L. R. et D. R. FULKERSON (1956). "Solving the Transportation Problem". In : *Management Science* 3.1, p. 24–32. DOI : 10.1287/mnsc.3.1.24.

GARG, N. et J. KONEMANN (1998). "Faster and simpler algorithms for multicommodity flow and other fractional packing problems". In : *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/sfcs.1998.743463.

GATZIANAS, Marios et Leonidas GEORGIADIS (2008). "A Distributed Algorithm for Maximum Lifetime Routing in Sensor Networks with Mobile Sink". In : *IEEE Transactions on Wireless Communications* 7.3, p. 984–994. DOI : 10.1109/twc.2008.060727.

GOLDBERG, D.E. (2006). *Genetic Algorithms*. Pearson Education. ISBN : 9788177588293.

GORDON, Basil *et al.* (1987). *Challenging mathematical problems with elementary solutions*. T. 1. Courier Corporation, p. 102–106.

GROSSO, Andrea, Marco LOCATELLI et Wayne PULLAN (2007). "Simple ingredients leading to very efficient heuristics for the maximum clique problem". In : *Journal of Heuristics* 14.6, p. 587–612. DOI : 10.1007/s10732-007-9055-x.

GU, Yu, Hengchang LIU et Baohua ZHAO (2007). "Target Coverage With QoS Requirements in Wireless Sensor Networks". In : *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/ipc.2007.116.

GU, Yu *et al.* (2009). "Target coverage problem in wireless sensor networks: A column generation based approach". In : *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/mobhoc.2009.5336962.

GU, Yu *et al.* (2011). "Theoretical Treatment of Target Coverage in Wireless Sensor Networks". In : *J. Comput. Sci. Technol.* 26.1, p. 117–129. DOI : 10.1007/s11390-011-9419-4.

GU, Yu *et al.* (2012a). "Covering Targets in Sensor Networks: From Time Domain to Space Domain". In : *IEEE Trans. Parallel Distrib. Syst.* 23.9, p. 1643–1656. DOI : 10.1109/tpds.2012.99.

— (2012b). "Towards an optimal lifetime in heterogeneous surveillance wireless sensor networks". In : *EURASIP J Wirel Commun Netw* 2012.1, p. 74. DOI : 10.1186/1687-1499-2012-74.

GUREVSKY, Evgeny (2011). "Conception de lignes de fabrication sous incertitudes : analyse de sensibilité et approche robuste." Thèse de doctorat dirigée par Dolgui, Alexandre Génie Industriel Saint-Etienne, EMSE 2011. Thèse de doct. URL : http://www.theses.fr/2011EMSE0634.

HANDY, M.J., M. HAASE et D. TIMMERMANN (2002). "Low energy adaptive clustering hierarchy with deterministic cluster-head selection". In : *4th International Workshop on Mobile and Wireless Communications Network*. Institute of Electrical & Electronics Engineers (IEEE), p. 368–372. DOI : 10.1109/mwcn.2002.1045790.

HANSEN, Pierre, Nenad MLADENOVIĆ et José A. Moreno PÉREZ (2009). "Variable neighbourhood search: methods and applications". In : *Annals of Operations Research* 175.1, p. 367–407. DOI : 10.1007/s10479-009-0657-6.

Henna, Shagufta et Thomas Erlebach (2013). "Approximating Maximum Disjoint Coverage in Wireless Sensor Networks". In : *Ad-hoc, Mobile, and Wireless Network*. Springer Science + Business Media, p. 148–159. doi : 10.1007/978-3-642-39247-4_13.

Huangfu, Wei *et al.* (2014). "Survivability-oriented optimal node density for randomly deployed wireless sensor networks". In : *Science China Information Sciences* 57.2, p. 1–6. doi : 10.1007/s11432-013-5054-8.

IBM ILOG CPLEX Optimizer. http : //www -01.ibm.com/software/integration/optimization/cplex-optimizer/.

Jameii, Seyed Mahdi, Karim Faez et Mehdi Dehghan (2015). "AMOF: adaptive multiobjective optimization framework for coverage and topology control in heterogeneous wireless sensor networks". In : *Telecommunication Systems* 61.3, p. 515–530. doi : 10.1007/s11235-015-0009-6.

Jeong, J. *et al.* (2007). "MCTA: Target Tracking Algorithm Based on Minimal Contour in Wireless Sensor Networks". In : *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. Institute of Electrical & Electronics Engineers (IEEE). doi : 10.1109/infcom.2007.283.

Jishy, Khalil (2011). "Detection and tracking of maneuvering targets on passive radar by Gaussian particles filtering". Theses. Institut National des Télécommunications. url : https://tel.archives-ouvertes.fr/tel-00697130.

Kaddour, Mejdi (2015). "Optimizing the throughput-lifetime tradeoff in wireless sensor networks with link scheduling, rate adaptation, and power control". In : *Wirel. Commun. Mob. Comput.* n/a–n/a. doi : 10.1002/wcm.2613.

Kouvelis, Panos et Gang Yu (1997). *Robust Discrete Optimization and Its Applications*. Springer US. doi : 10.1007/978-1-4757-2620-6.

Kung, H.T. et D. Vlah (2003). "Efficient location tracking using sensor networks". In : *IEEE Wireless Communications and Networking (WCNC)*. T. 3. Institute of Electrical & Electronics Engineers (IEEE), p. 1954–1961. doi : 10.1109/wcnc.2003.1200686.

Lee, Cheng-Ta, Frank Yeong-Sung Lin et Yean-Fu Wen (2006). "An Efficient Object Tracking Algorithm in Wireless Sensor Networks". In : *Proceedings of the 9th Joint Conference on Information Sciences (JCIS)*. Atlantis Press. doi : 10.2991/jcis.2006.207.

Lenzen, C. et R. Wattenhofer (2011). "Distributed algorithms for sensor networks". In : *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 370.1958, p. 11–26. doi : 10.1098/rsta.2011.0212.

Lersteau, Charly, André Rossi et Marc Sevaux (2016). “Robust scheduling of wireless sensor networks for target tracking under uncertainty”. In : *European Journal of Operational Research* 252.2, p. 407–417. doi : 10.1016/j.ejor.2016.01.018.

Lin, Frank Yeong-Sung *et al.* (2011). “Evaluation of Network Survivability Considering Degree of Disconnectivity”. In : *Lecture Notes in Computer Science*. Springer Science + Business Media, p. 51–58. doi : 10.1007/978-3-642-21219-2_8.

Lin, Frank Yeong-Sung, Pei-Yu Chen et Quen-Ting Chen (2012). “Resource Allocation Strategies to Maximize Network Survivability Considering of Average DOD”. In : *Advances in Intelligent and Soft Computing*. Springer Science + Business Media, p. 751–758. doi : 10.1007/978-3-642-28765-7_90.

Liu, Hai *et al.* (2011). “General Maximal Lifetime Sensor-Target Surveillance Problem and Its Solution”. In : *IEEE Trans. Parallel Distrib. Syst.* 22.10, p. 1757–1765. doi : 10.1109/tpds.2011.42.

Liu, Li *et al.* (2009). “Heuristics for Mobile Object Tracking Problem in Wireless Sensor Networks”. In : *Frontiers in Algorithmics*. Springer Science + Business Media, p. 251–260. doi : 10.1007/978-3-642-02270-8_26.

Lübbecke, Marco E. et Jacques Desrosiers (2005). “Selected Topics in Column Generation”. In : *Operations Research* 53.6, p. 1007–1023. doi : 10.1287/opre.1050.0234.

Madan, R. et S. Lall (2006). “Distributed algorithms for maximum lifetime routing in wireless sensor networks”. In : *IEEE Transactions on Wireless Communications* 5.8, p. 2185–2193. doi : 10.1109/twc.2006.1687734.

Mahmood, Muhammad Adeel, Winston K.G. Seah et Ian Welch (2015). “Reliability in wireless sensor networks: A survey and challenges ahead”. In : *Computer Networks* 79, p. 166–187. doi : 10.1016/j.comnet.2014.12.016.

Martins, Flávio V. C. *et al.* (2011). “A Hybrid Multiobjective Evolutionary Approach for Improving the Performance of Wireless Sensor Networks”. In : *IEEE Sensors J.* 11.3, p. 545–554. doi : 10.1109/jsen.2010.2048897.

Martins, Flávio VC *et al.* (2007). “Model and algorithms for the density, coverage and connectivity control problem in flat WSNs”. In : *Proceedings of the International Network Optimization Conference (INOC’07)*, p. 1145–1152.

Meguerdichian, Seapahn et Miodrag Potkonjak (2003). “Low power 0/1 coverage and scheduling techniques in sensor networks”. In : *University of California, Los Angeles, Tech. Rep* 30001.

NADERAN, Marjan, Mehdi DEHGHAN et Hossein PEDRAM (2009). "Mobile object tracking techniques in wireless sensor networks". In : *2009 International Conference on Ultra Modern Telecommunications & Workshops*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/icumt.2009.5345626.

NADERAN, Marjan *et al.* (2012). "Survey of mobile object tracking protocols in wireless sensor networks: a network-centric perspective". In : *IJAHUC* 11.1, p. 34. DOI : 10.1504/ijahuc.2012.049283.

ORACEVIC, Alma et Suat OZDEMIR (2014). "A survey of secure target tracking algorithms for wireless sensor networks". In : *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/wccais.2014.6916628.

ORLIN, James B. (2013). "Max flows in O(nm) time, or better". In : *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13*. Association for Computing Machinery (ACM). DOI : 10.1145/2488608.2488705.

ÖZDEMIR, Suat, Bara'a A. ATTEA et Önder A. KHALIL (2012a). "Multi-objective clustered-based routing with coverage control in wireless sensor networks". In : *Soft Comput* 17.9, p. 1573–1584. DOI : 10.1007/s00500-012-0970-x.

— (2012b). "Multi-Objective Evolutionary Algorithm Based on Decomposition for Energy Efficient Coverage in Wireless Sensor Networks". In : *Wireless Pers Commun* 71.1, p. 195–215. DOI : 10.1007/s11277-012-0811-3.

PATEL, Dipesh J., Rajan BATTA et Rakesh NAGI (2005). "Clustering Sensors in Wireless Ad Hoc Networks Operating in a Threat Environment". In : *Operations Research* 53.3, p. 432–442. DOI : 10.1287/opre.1040.0171.

PINHEIRO, Placido Rogerio, Álvaro Meneses Sobreira NETO et Alexei Barbosa AGUIAR (2013). "Handing Optimization Energy Consumption in Heterogeneous Wireless Sensor Networks". In : *International Journal of Distributed Sensor Networks* 2013, p. 1–9. DOI : 10.1155/2013/328619.

RAICONI, Andrea et Monica GENTILI (2011). "Exact and Metaheuristic Approaches to Extend Lifetime and Maintain Connectivity in Wireless Sensors Networks". In : *Lecture Notes in Computer Science*. Springer Science + Business Media, p. 607–619. DOI : 10.1007/978-3-642-21527-8_68.

ROSSI, André, Alok SINGH et Marc SEVAUX (2010). "A column generation scheme for a collection of sensor network scheduling problems". In : *Proceedings of 11ème Congrés de*

la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Toulouse, France.

Rossi, André, Alok Singh et Marc Sevaux (2011). "Column generation algorithm for sensor coverage scheduling under bandwidth constraints". In : *Networks* 60.3, p. 141–154. doi : 10.1002/net.20466.

— (2012). "An exact approach for maximizing the lifetime of sensor networks with adjustable sensing ranges". In : *Computers & Operations Research* 39.12, p. 3166–3176. doi : 10.1016/j.cor.2012.04.001.

— (2013). "Lifetime maximization in wireless directional sensor network". In : *European Journal of Operational Research* 231.1, p. 229–241. doi : 10.1016/j.ejor.2013.05.033.

Santos, Andréa Cynthia *et al.* (2012). "Strategies for designing energy-efficient clusters-based WSN topologies". In : *Journal of Heuristics* 18.4, p. 657–675. doi : 10.1007/s10732-012-9202-x.

Sastry, Kumara, David E. Goldberg et Graham Kendall (2013). "Genetic Algorithms". In : *Search Methodologies*. Springer Science + Business Media, p. 93–117. doi : 10.1007/978-1-4614-6940-7_4.

Sengupta, S. *et al.* (2012). "An Evolutionary Multiobjective Sleep-Scheduling Scheme for Differentiated Coverage in Wireless Sensor Networks". In : *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6, p. 1093–1102. doi : 10.1109/tsmcc.2012.2196996.

Sengupta, Soumyadip *et al.* (2013). "Multi-objective node deployment in WSNs: In search of an optimal trade-off among coverage, lifetime, energy consumption, and connectivity". In : *Engineering Applications of Artificial Intelligence* 26.1, p. 405–416. doi : 10.1016/j.engappai.2012.05.018.

Shi, Hai-Yan *et al.* (2012). "Game Theory for Wireless Sensor Networks: A Survey". In : *Sensors* 12.12, p. 9055–9097. doi : 10.3390/s120709055.

Singh, Alok et André Rossi (2013a). "A genetic algorithm based exact approach for lifetime maximization of directional sensor networks". In : *Ad Hoc Networks* 11.3, p. 1006–1021. doi : 10.1016/j.adhoc.2012.11.004.

Singh, Alok, André Rossi et Marc Sevaux (2013b). "Matheuristic approaches for Q - coverage problem versions in wireless sensor networks". In : *Engineering Optimization* 45.5, p. 609–626. doi : 10.1080/0305215x.2012.687732.

Slijepcevic, S. et M. Potkonjak (2001). “Power efficient organization of wireless sensor networks”. In : *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*. Institute of Electrical & Electronics Engineers (IEEE). doi : 10.1109/icc.2001.936985.

Sotskov, Yuri N., Albert P.M. Wagelmans et Frank Werner (1998). “On the Calculation of the Stability Radius of an Optimal or an Approximate Schedule”. In : *Annals of Operations Research* 83, p. 213–252. doi : 10.1023/a:1018960030420.

Sotskov, Yuri N., Alexandre Dolgui et Marie-Claude Portmann (2006). “Stability analysis of an optimal balance for an assembly line with fixed cycle time”. In : *European Journal of Operational Research* 168.3, p. 783–797. doi : 10.1016/j.ejor.2004.07.028.

Tian, Jie *et al.* (2015). “Scheduling Survivability-Heterogeneous Sensor Networks for Critical Location Surveillance”. In : *ACM Trans. Sen. Netw.* 11.4, p. 1–23. doi : 10.1145/2764914.

Türkoğullari, Yavuz Boğaç *et al.* (2007). “Optimal placement and activity scheduling to maximize coverage lifetime in wireless sensor networks”. In : *2007 22nd international symposium on computer and information sciences*. Institute of Electrical & Electronics Engineers (IEEE). doi : 10.1109/iscis.2007.4456874.

— (2010a). “A column generation based heuristic for sensor placement, activity scheduling and data routing in wireless sensor networks”. In : *European Journal of Operational Research* 207.2, p. 1014–1026. doi : 10.1016/j.ejor.2010.05.020.

— (2010b). “An efficient heuristic for placement, scheduling and routing in wireless sensor networks”. In : *Ad Hoc Networks* 8.6, p. 654–667. doi : 10.1016/j.adhoc.2010.01.005.

Wang, Chen *et al.* (2008). “Optimization scheme for sensor coverage scheduling with bandwidth constraints”. In : *Optimization Letters* 3.1, p. 63–75. doi : 10.1007/s11590-008-0091-8.

Wang, Lan et Yang Xiao (2006). “A Survey of Energy-Efficient Scheduling Mechanisms in Sensor Networks”. In : *Mobile Networks and Applications* 11.5, p. 723–740. doi : 10.1007/s11036-006-7798-5.

Wang, Yu-Shun *et al.* (2013). “Maximization of Wireless Mesh Networks Survivability to Assure Service Continuity under Intelligent Attacks”. In : *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. Institute of Electrical & Electronics Engineers (IEEE). doi : 10.1109/aina.2013.31.

WARD, James E. et Richard E. WENDELL (1990). “Approaches to sensitivity analysis in linear programming”. In : *Annals of Operations Research* 27.1, p. 3–38. DOI : 10.1007/bf02055188.

“Wireless Sensor Networks See Uptake in Global Industries Due to Technological Breakthroughs” (2015). In : *Frost & Sullivan*. URL : http://ww2.frost.com/news/press-releases/wireless-sensor-networks-see-uptake-global-industries-due-technological-breakthroughs/.

XU, Yingqi, J. WINTER et Wang-Chien LEE (2004). “Prediction-based strategies for energy saving in object tracking sensor networks”. In : *IEEE International Conference on Mobile Data Management, 2004. Proceedings. 2004.* Institute of Electrical & Electronics Engineers (IEEE), p. 346–357. DOI : 10.1109/mdm.2004.1263084.

YANG, H. et B. SIKDAR (2003). “A protocol for tracking mobile targets using sensor networks”. In : *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.* Institute of Electrical & Electronics Engineers (IEEE), p. 71–78. DOI : 10.1109/snpa.2003.1203358.

YANG, H., D. LI et H. CHEN (2010). “Coverage Quality Based Target-Oriented Scheduling in Directional Sensor Networks”. In : *2010 IEEE International Conference on Communications*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/icc.2010.5501996.

YEONG-SUNG, Frank, CHENG-TA et Yen-Yi HSU (2010). “An energy-efficient algorithm for object tracking in Wireless Sensor Networks”. In : *2010 IEEE International Conference on Wireless Communications, Networking and Information Security*. Institute of Electrical & Electronics Engineers (IEEE). DOI : 10.1109/wcins.2010.5544123.

YICK, Jennifer, Biswanath MUKHERJEE et Dipak GHOSAL (2008). “Wireless sensor network survey”. In : *Computer Networks* 52.12, p. 2292–2330. DOI : 10.1016/j.comnet.2008.04.002.

YOUNIS, O. et S. FAHMY (2004). “HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks”. In : *IEEE Transactions on Mobile Computing* 3.4, p. 366–379. DOI : 10.1109/tmc.2004.41.

ZHANG, W. et G. CAO (2004). “DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks”. In : *IEEE Transactions on Wireless Communications* 3.5, p. 1689–1701. DOI : 10.1109/twc.2004.833443.

Zhao, Miao et Yuanyuan Yang (2012). "Optimization-Based Distributed Algorithms for Mobile Data Gathering in Wireless Sensor Networks". In : *IEEE Transactions on Mobile Computing* 11.10, p. 1464–1477. doi : 10.1109/tmc.2011.178.

Zhao, Qun et M. Gurusamy (2008). "Lifetime Maximization for Connected Target Coverage in Wireless Sensor Networks". In : *IEEE/ACM Transactions on Networking* 16.6, p. 1378–1391. doi : 10.1109/tnet.2007.911432.

Zu, Chen-Xi et Hong Li (2011). "Thermodynamic analysis on energy densities of batteries". In : *Energy & Environmental Science* 4.8, p. 2614. doi : 10.1039/c0ee00777c.

Table des figures

Liste des tableaux

Liste des Algorithmes

Liste des modèles

Optimisation de réseaux de capteurs sans fil pour le suivi de cibles mobiles

Charly Lersteau

Résumé

Les réseaux de capteurs sans fil suscitent une attention croissante depuis quelques années, tant les applications sont nombreuses, incluant notamment le suivi de véhicules ou la surveillance de champs de bataille. Un ensemble de capteurs disséminé aléatoirement a pour but de surveiller des cibles se déplaçant dans une région donnée. Chaque capteur a une durée de vie limitée et deux états : actif ou inactif. Un capteur actif peut surveiller des cibles dans son rayon de portée, au prix d'une consommation d'énergie. Dans cette thèse, les problèmes étudiés consistent à déterminer un ordonnancement optimal d'activités de surveillance, afin de couvrir toutes les cibles à tout instant de la mission.

Nous abordons en premier lieu un problème d'ordonnancement robuste. Une cible dont on connaît la trajectoire spatiale avec précision est sujette à incertitude temporelle. Cette situation est rencontrée lorsqu'un avion vole dans un couloir aérien, qu'un train circule sur une voie ferrée, ou que de tout autre véhicule suit un itinéraire pré-déterminé. L'objectif est de calculer un ordonnancement d'activités capable de résister au plus grand écart par rapport aux dates prévisionnelles de passage de la cible. Ce premier problème est résolu à l'aide d'un algorithme exact pseudo-polynomial, reposant sur une dichotomie.

En second lieu, nous étudions le problème visant à préserver la capacité de surveillance du réseau de capteurs dans un contexte multi-missions. Les cibles sont maintenant sujettes à une incertitude spatiale, c'est-à-dire susceptibles de se trouver à une distance inférieure à un seuil δ de leur position prévisionnelle. Ce second problème est résolu à l'aide d'un algorithme exact basé sur la génération de colonnes, et accéléré par une métaheuristique.

Les méthodes de résolution proposées ont en commun une étape préliminaire, appelée discrétisation, qui conduit à reformuler les problèmes originaux comme des problèmes d'ordonnancement d'activités de surveillance. L'espace de surveillance est découpé en faces, ensembles de points couverts par un même sous-ensemble de capteurs. Le calcul des durées de séjour des cibles dans chaque face permet de découper la durée de la mission en fenêtres de temps, et d'envisager le problème de couverture de cibles mobiles comme une séquence de problèmes de couverture de cibles immobiles.

Les algorithmes proposés pour aborder ces problèmes sont testés sur de nombreuses instances, et leurs résultats sont analysés. De nombreuses perspectives ouvertes par ces travaux sont également présentées.

Abstract

Wireless sensor networks have received a particular attention during the last years, involving many applications, such as vehicle tracking or battlefield monitoring. A set of sensors is randomly dispatched in a region in order to monitor moving targets. Each sensor has a limited battery lifetime and two states: active or inactive. An active sensor is able to monitor targets inside its sensing radius, which consumes energy. In this thesis, the studied problems consist in deciding an optimal schedule of sensing activities, in order to cover all the targets at any instant of the mission.

First, we study a robust scheduling problem. A target such that the spatial trajectory is exactly known is subject to temporal uncertainties. This context is met for a plane flying in an airline route, a train running on a railway, or any vehicle following a predetermined path. The objective is to compute a schedule of activities able to resist to the largest uncertainties. This first problem is solved using an exact pseudo-polynomial algorithm, relying on a dichotomy.

Second, we study a problem aiming at preserving enough sensor network capacity in order to perform further missions. For this problem, the targets are subject to spatial uncertainties, i.e. their actual position may be at a distance δ of their expected position. This second problem is solved using an exact algorithm based on column generation, accelerated by a metaheuristic.

All the proposed methods have a common phase, called discretization, that leads to reformulate the original problems as activity scheduling problems. The monitored area is split into faces, that are defined as sets of points covered by the same set of sensors. Computing the stay duration of targets inside each face leads to split the mission duration into time windows, so the moving target tracking problem can be seen as a sequence of static target tracking problems.

The proposed algorithms are tested on many instances, and the analysis of the results is provided. Numerous open perspectives of this work are also given.